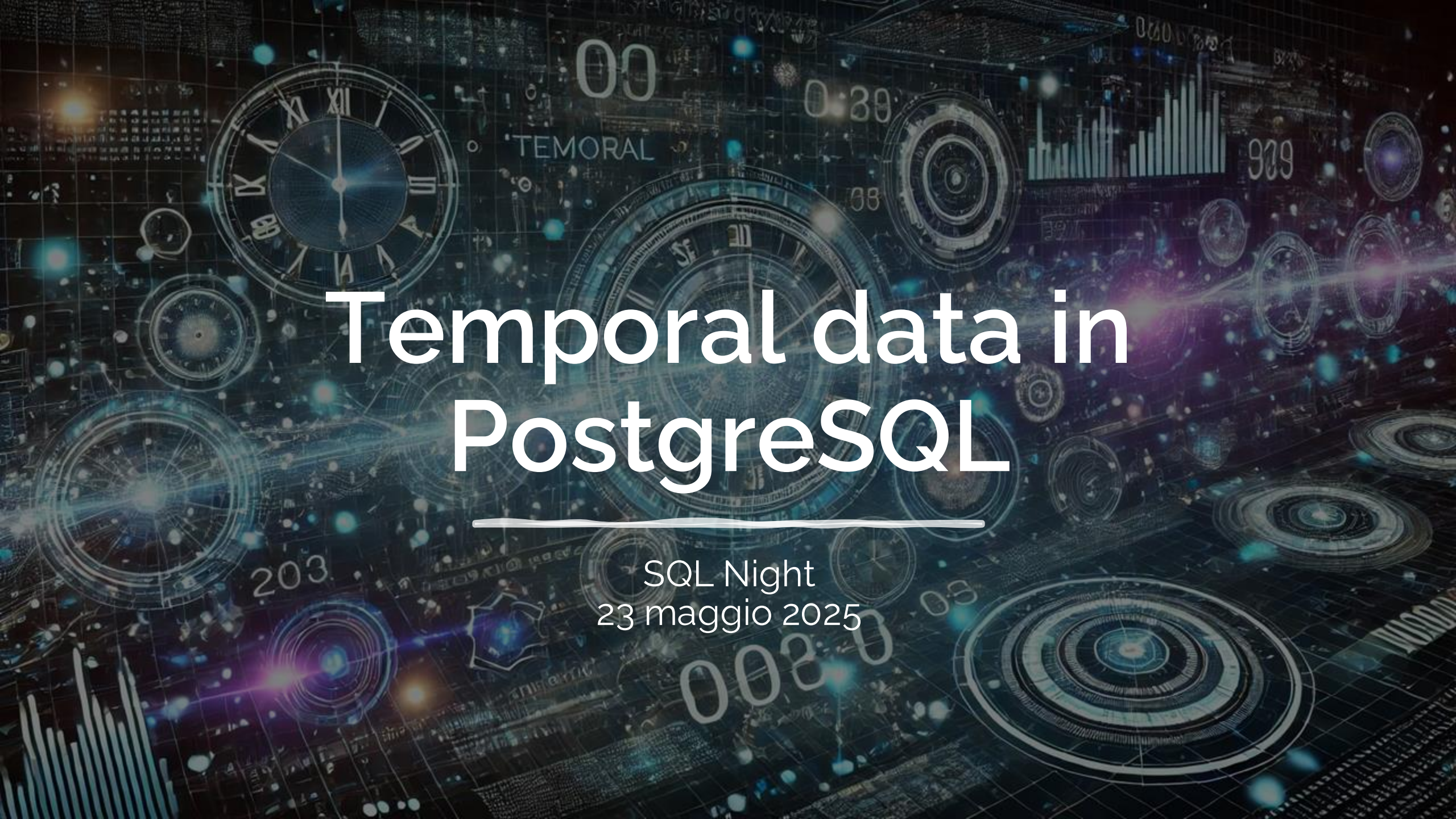

Temporal data in PostgreSQL

SQL Night
23 maggio 2025

Chi è Danilo Dominici

- Consulente Senior : database
  - SQL Server
  - PostgreSQL
  - Redis
- Creatore di SQL Start! (www.sqlstart.it) – evento community (Ancona, 13 giugno 2025)
- Community speaker, MS Certified Trainer dal 2000, 6x Data Platform MVP

Microsoft CERTIFIED TRAINER MCT

2000+

MVP Microsoft® Most Valuable Professional

2014-2020

Tabelle temporali

Tabelle temporali

- Strutture-dati che **conservano in automatico lo stato passato di ogni riga** e permettono query "time-travel" senza dover gestire manualmente tabelle di audit o trigger
- Lo standard ISO/IEC 9075 ne definisce due assi temporali:

Asse	Significato	Colonne tipiche	Sinonimi comuni
Valid-time	Quando il dato è (o era) "vero" nel dominio applicativo	VALID_FROM, VALID_TO	"application time"
Transaction-time	Quando il dato è stato fisicamente scritto nel DB	SYS_START, SYS_END	"system time"

- Una tabella può implementare uno dei due assi (mono-temporale) o entrambi (bi-temporale).

Tabelle temporali

- **Coerenza ACID**: il salvataggio della riga attuale e della versione storica avviene nella stessa transazione.
- **Integrità temporale**: il motore impedisce sovrapposizioni nei periodi.
- **Indicizzabilità**: è possibile indicizzare colonne normali e quelle di periodo.
- **Gestione del retention**: policy per troncare automaticamente la history
- **Compatibilità con replica e CDC**: le versioni storiche viaggiano con la tabella principale.
- **Limiti**: – niente TRUNCATE; – alcune opzioni di ALTER TABLE richiedono la disattivazione temporanea del versioning; – attenzione allo spazio su disco (la history può crescere rapidamente).

Scenari di utilizzo tipici

Scenario	Perché le tabelle temporali sono utili
Audit e tracciabilità SOX/GDPR	Log automatico di chi ha cambiato cosa e quando, senza codice extra.
Ricostruzione di report retrospettivi	Ricreare il bilancio "come appariva il 31 dicembre" anche dopo correzioni successive.
Slowly Changing Dimension (SCD Tipo 2)	Nei data-warehouse basta una sola tabella per tenere versioni storiche dei master data.
Debug di incidenti	"Il valore era giusto ieri?" – basta una query AS OF.
Analisi bitemporale	Studiare discrepanze fra "quando il fatto è avvenuto" e "quando è stato registrato" (es. ritardi di back-office).

Scenari di utilizzo tipici

Scenario	Perché le tabelle temporali sono utili
Ricalcoli fiscali o assicurativi	Norme o tariffe retroattive: applico la versione del dato valida in quel periodo.
Right to rectification (GDPR art. 16)	La rettifica genera una nuova riga con nuova valid-time; la storia precedente rimane immutabile (accountability).
Machine Learning basato su serie storiche	Estraggo dataset al volo per qualsiasi finestra temporale senza snapshot manuali.
Simulazioni "what-if"	Copio una versione storica in ambiente di test per stress-test o migrazioni.
Event sourcing light	In sistemi OLTP tradizionali posso ri-creare l'evento (insert/update/delete) dal diff fra versioni.

PostgreSQL e i dati temporali

PostgreSQL standard: cosa c'è

- PostgreSQL non integra ancora il supporto nativo allo standard: è oggetto di discussione ma non è ancora stato implementato nella v17
- Si possono però ottenere funzioni equivalenti tramite:
  - **Estensioni** C o PL/pgSQL
  - **Trigger + History table**: al verificarsi di AFTER INSERT/UPDATE/DELETE si sposta la riga manualmente
  - **Range type + exclusion constraint**: Colonna di tipo tstzrange per il periodo e un indice GiST con esclusione per evitare sovrapposizioni
  - **MVCC system columns** (xmin e xmax): utile per il debug, ma non per conservazione a lungo termine (VACUUM li reinizializza)

PostgreSQL standard: cosa manca

Limite	Commento
Mancanza di sintassi FOR SYSTEM_TIME	Le query richiedono filtri espliciti (WHERE TIMESTAMP '2025-05-01' <@ period).
Nessun DDL automatico per ALTER TABLEcon history	Se aggiungi una colonna, va replicata manualmente nella _hist.
Chiavi esterne temporali	Non c'è enforcement nativo (temporal FK) – occorre logica custom. (The World of Data)
Riutilizzo di xmin	Non affidabile per archiviazioni oltre il ciclo di VACUUM.
Estensioni non disponibili su alcuni managed services	Su RDS/Aurora o Cloud SQL servono versioni in puro PL/pgSQL. (GitHub)

Demo

Le estensioni per i dati temporali

Le principali estensioni per i dati temporali

Extension	Temporal Dimensions	Impl. & Cloud-Friendliness	Last Release (Stars)	Pros	Cons
temporal_tables	System-time only	C extension (needs superuser)	v1.2.2 · 960★	Fast, mature, simple versioning() trigger	Not valid-time; blocks on RDS/Azure
periods	Uni-, bi-, tri-temporal (SQL 2016 syntax)	C extension	v1.2 · 290★	Near-standard SQL syntax; supports WITHOUT OVERLAPS constrai nts	Early-stage; limited docs; no managed-DB support
temporal_tables _plpgsql (NearForm rewrite)	System-time	Pure PL/pgSQL (works on RDS/Azure)	2024-02 tag	Cloud-friendly; no compile step	~2× slower than C; fewer guard rails

Le principali estensioni per i dati temporali

Extension	Temporal Dimensions	Impl. & Cloud-Friendliness	Last Release (Stars)	Pros	Cons
table_version	Snapshot-by-revision (manual)	PL/pgSQL + triggers	v1.4 · 180★	Fine-grained revision API , diff functions	Requires explicit revision calls; no automatic period logic
pgmemento	System-time + DDL history	PL/pgSQL (cloud OK)	v0.7 · 390★	Tracks DML **and** DDL; JSONB delta log; restore helpers	Heavier schema; no valid-time; learning curve
pg_bitemporal	Full bi-temporal	PL/pgSQL, relies on GiST/exclusion	2023-11 · 150★	True application + system time; constraint-based integrity	Less active; no automated history partitioning

temporal_tables

- System-time only
- C extension (requires superuser)
- Fast, simple versioning()
- Not valid-time; not on RDS/Azure

periods

- Uni/Bi/Tri-temporal
- C extension
- Near-standard SQL syntax
- Early-stage; limited docs

temporal_tables_plpgsql

- System-time
- Pure PL/pgSQL
- Cloud-friendly
- ~2× slower than C version

table_version

- Snapshot-by-revision
- PL/pgSQL + triggers
- Diff functions
- Requires explicit revision calls

pgmemento

- System-time + DDL history
- PL/pgSQL
- Tracks DML & DDL
- Heavier schema; learning curve

pg_bitemporal

- Full bi-temporal
- PL/pgSQL, GiST/exclusion
- True application + system-time
- Less active maintenance
- Started in 2015, from 2020 "forked" and maintained by Hettie Dombrovskaya
- https://github.com/hettie-d/pg_bitemporal

Demo

Considerazioni finali

- Utilizzare indici BRIN per le tabelle storiche
- Usare le esclusioni negli indici GiST per gli *overlaps*
- Partizionare i dati storici per anno o anno/mese
- La tabella storico cresce rapidamente: regolare la soglia di AUTOVACUUM
- Utilizzare DEFERRABLE INITIALLY IMMEDIATE per garantire la coerenza anche con transazioni lunghe

Resources

- https://sigmodrecord.org/publications/sigmodRecord/1209/pdfs/07.industry.kulkarni.pdf
- https://www.wiscorp.com/SQLStandards.html
- https://www.red-gate.com/simple-talk/author/borisnovikov/

Resources

- PERIODS
  https://github.com/xocolatl/periods
- TEMPORAL_TABLES
  https://github.com/nearform/temporal_tables
- BI-TEMPORAL
  https://github.com/hettie-d/pg_bitemporal/blob/master/README.md

Q & A tempo permettendo... ☺

Altrimenti -> ddominici@gmail.com

Grazie!