

POSTGRESQL

POSTGRESQL come sistema transazionale

Michele Finelli
m@biodec.com
BioDec

Indice

Cos'è un sistema transazionale

La questione dell'isolamento

La questione della persistenza

Come POSTGRESQL è transazionale

Repliche

?

Indice

Cos'è un sistema transazionale

La questione dell'isolamento

La questione della persistenza

Come POSTGRESQL è transazionale

Repliche

?

Punti di forza di PostgreSQL

POSTGRESQL è un database server con le seguenti salienti caratteristiche:

1. sistema **transazionale**,
2. funzionalità di *high availability* e di replica,
3. sistema *enterprise*,
4. *free software*,
5. gratuità.

ACID

1. *ACID* significa *Atomicity, Consistency, Isolation e Durability*.
2. In un database ACID tali proprietà valgono sempre.
3. Se esistono delle situazioni in cui anche una sola delle proprietà cessa di valere, il sistema è *ipso facto* **non transazionale**.

Non essere ACID

- ▶ Si noti che non essere ACID non è di per sé un problema: esistono numerosi componenti di un sistema che **non sono transazionali**: ad esempio un *file system*.
- ▶ Il problema è che garantire le proprietà di integrità e di persistenza richieste è **costoso**, in termini di risorse computazionali — e per certi versi impossibile se trattiamo sistemi distribuiti.
- ▶ La quasi totalità dei sistemi NoSQL è in realtà **NoTransazionale** (il fatto che il *query language* non sia SQL è spesso un effetto collaterale).

ACID: an ill defined concept

1. *Atomicity* ... non è proprio *atomicità*,
2. *Consistency* ... lo è a modo suo,
3. di *Isolation levels* ce ne sono una pletera ...
4. se vogliamo, anche *Durability* di solito si chiama *Persistence*.

*ACID: more mnemonic than precise – Eric Brewer,
2012*

Isolamento e persistenza

- ▶ Nel seguito della presentazione ci focalizzeremo su due aspetti soli dell'essere ACID: l'isolamento e la persistenza.
- ▶ Vedremo prima cosa essi comportano da un punto di vista teorico.
- ▶ E successivamente come tali questioni sono affrontate in POSTGRESQL.

Tipi di anomalie

Anomalia	Descrizione	Bruttezza
<i>Dirty read</i>	Una transazione accede ad un dato scritto da una transazione concorrente, la quale non sia stata ancora committata .	TOTALE
<i>Non-repeatable read</i>	Una transazione ripetendo una lettura trova un dato che è stato modificato da una transazione che è terminata nel mentre.	Marcata
<i>Phantom read</i>	Come sopra, ma ripetendo una query e trovando un diverso insieme di righe.	<i>mmmh ...</i>

Anomalie consentite = livello di isolamento

- ▶ Lo standard SQL definisce quattro livelli di isolamento delle transazioni, così chiamati:
 - ▶ *Read Uncommitted*,
 - ▶ *Read Committed*,
 - ▶ *Repeatable Read*,
 - ▶ *Serializable*.
- ▶ Ogni livello permette o meno il presentarsi di una o più delle anomalie appena viste:
 - ▶ Dirty read,
 - ▶ Non-repeatable read,
 - ▶ Phantom read.

Livelli di isolamento

Livello di isolamento	Anomalia consentita		
	Dirty read	Non-repeatable read	Phantom read
Read uncommitted	Si	Si	Si
Read committed	No	Si	Si
Repeatable read	No	No	Si
Serializable	No	No	No

From the source

*In PostgreSQL, you can request any of the four standard transaction isolation levels. But internally, **there are only three distinct isolation levels**, which correspond to the levels Read Committed, Repeatable Read, and Serializable.*

*When you select the level Read Uncommitted you really get Read Committed, and **phantom reads are not possible in the PostgreSQL implementation of Repeatable Read**, so the actual isolation level might be stricter than what you select.*

Indice

Cos'è un sistema transazionale

La questione dell'isolamento

La questione della persistenza

Come POSTGRESQL è transazionale

Repliche

?

fsync()

The fsync() function is intended to force a physical write of data from the buffer cache, and to assure that after a system crash or other failure that all data up to the time of the fsync() call is recorded on the disk. — POSIX

È un'operazione molto dispendiosa, perché forza un I/O altrimenti differibile.

Indice

Cos'è un sistema transazionale

La questione dell'isolamento

La questione della persistenza

Come PostgreSQL è transazionale

Repliche

?

ACID in PostgreSQL: MVCC + WAL

- ▶ Poiché PostgreSQL è un sistema transazionale, deve implementare tutte le proprietà ACID.
- ▶ La parte **ACI** di **ACID** è garantita da un meccanismo noto come *MVCC (Multi Version Concurrency Control)*.
- ▶ La parte **D** di **ACID** è garantita da un meccanismo diverso, noto come **WAL (Write Ahead Log)**.

MVCC

1. Quando una transazione inizia un'operazione di scrittura viene definita una *transaction id*, o *XID*.
2. La XID è un valore a 32 bit.

MVCC

3. La **XID** determina la visibilità della transazione: tutte le transazioni con **XID** minore di una **XID** t sono nel **passato di t** , tutte quelle con **XID** maggiore sono nel **futuro di t** .
4. Le transazioni nel passato sono **visibili**, quelle nel futuro sono **invisibili**.

MVCC

5. I controlli dell'MVCC sono fatti al livello della singola tupla — per semplificare: l'unità di storage minima — con due campi `xmin` e `xmax`.
6. Quando una tupla viene creata `xmin` è posto pari all'XID della transazione che l'ha creata.
7. Quando una tupla viene cancellata, `xmax` è posto pari all'XID della transazione che l'ha cancellata.

WAL

La parte **D** di **ACID** è garantita da un meccanismo diverso, noto come *WAL* (Write Ahead Log).

1. In pratica, quando una *data page* è aggiornata, la modifica è immediatamente salvata sullo storage, nel WAL appunto.
2. Il data file corrispondente è modificato in seguito.
3. In caso di *crash* del database, il WAL viene letto e in caso di inconsistenza viene applicato nuovamente sui data file.

Database block lifecycle

4. La transazione scrive sul WAL la modifica. Da questo momento che il blocco dirty sia poi scritto sullo storage è secondario al fine di ripristinare, ad esempio in caso di *failure*, l'informazione che il blocco sia stato cambiato.
5. Le statistiche in `pg_stat_user_tables` sono aggiornate in modo asincrono.
6. Sempre in modo asincrono, altri processi si preoccupano di scrivere il blocco dirty anche sul disco, e a quel punto l'area di memoria è marcata *clean*.

Indice

Cos'è un sistema transazionale

La questione dell'isolamento

La questione della persistenza

Come POSTGRESQL è transazionale

Repliche

?

Replica basata sul log shipping

- ▶ Le soluzioni di replica fra più istanze di POSTGRESQL utilizzano una tecnica, detta *Transaction Log Shipping*.
- ▶ Essa consiste nel mantenere dei server *in standby* aggiornati con il master attraverso la copia del WAL.
- ▶ Nonostante la tecnica sia unica, vi sono più soluzioni possibili, che si distinguono per essere basate sulla copia fisica del WAL file, o sulla copia continua (detta anche *streaming replication*) o una combinazione di entrambe le modalità.

Server standby e HA

- ▶ Un *server standby* è un server che riceve il WAL archiviato appena questo è disponibile e lo applica.
- ▶ Il server in standby può essere configurato anche come *hot standby* per ricevere le *query* di lettura.

Streaming replication

- ▶ Se non c'è attività sul server principale, l'operazione di archivio del WAL può essere differita, e quindi portare a ritardi nella sincronizzazione fra il server principale e quello in standby.
- ▶ Esiste un parametro `archive_timeout` che può forzare uno *switch* del WAL dopo un certo periodo.
- ▶ È però preferibile un'altra modalità di replica, detta appunto *streaming replication*, in cui il WAL viene copiato sul server in standby man mano che questi si crea.



Indice

Cos'è un sistema transazionale

La questione dell'isolamento

La questione della persistenza

Come POSTGRESQL è transazionale

Repliche

?

Conclusioni

- ▶ PostgreSQL è un sistema transazionale **vero**.
- ▶ Esso ottiene tale obiettivo con un'implementazione **MVCC** che ha delle particolarità, e con un meccanismo di **WAL** per quanto riguarda la *Durability*.
- ▶ Avendo un meccanismo di WAL, una tecnica di *shipping* del medesimo è alla base della funzionalità di **replica** fra più istanze di PostgreSQL.
- ▶ Avere diverse repliche permette di realizzare sistemi *fault tolerant*, scalabili, eccetera.



Domande



Next

- ▶ La licenza è della presentazione è CC-BY-SA4.0 a.k.a. “Bàn xa vút da la vétta”; il PDF è reperibile all’URL
<https://github.com/finelli/slideware>
- ▶ Contatti: m@biodec.com
- ▶ Ci vediamo all’Incontro DevOps Italia 2016, 1 Aprile 2016 a.k.a. “Non è uno scherzo”, a Bologna — maggiori informazioni su
<http://www.incontrodevops.it/>