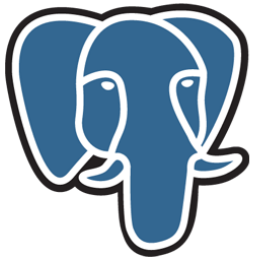


PostgreSQL

PGDay.IT 2011
Monash University Prato Centre
Venerdì 25 Novembre 2011

LOB vs Bytea

Carlo Ascani
Italian PostgreSQL Users Group

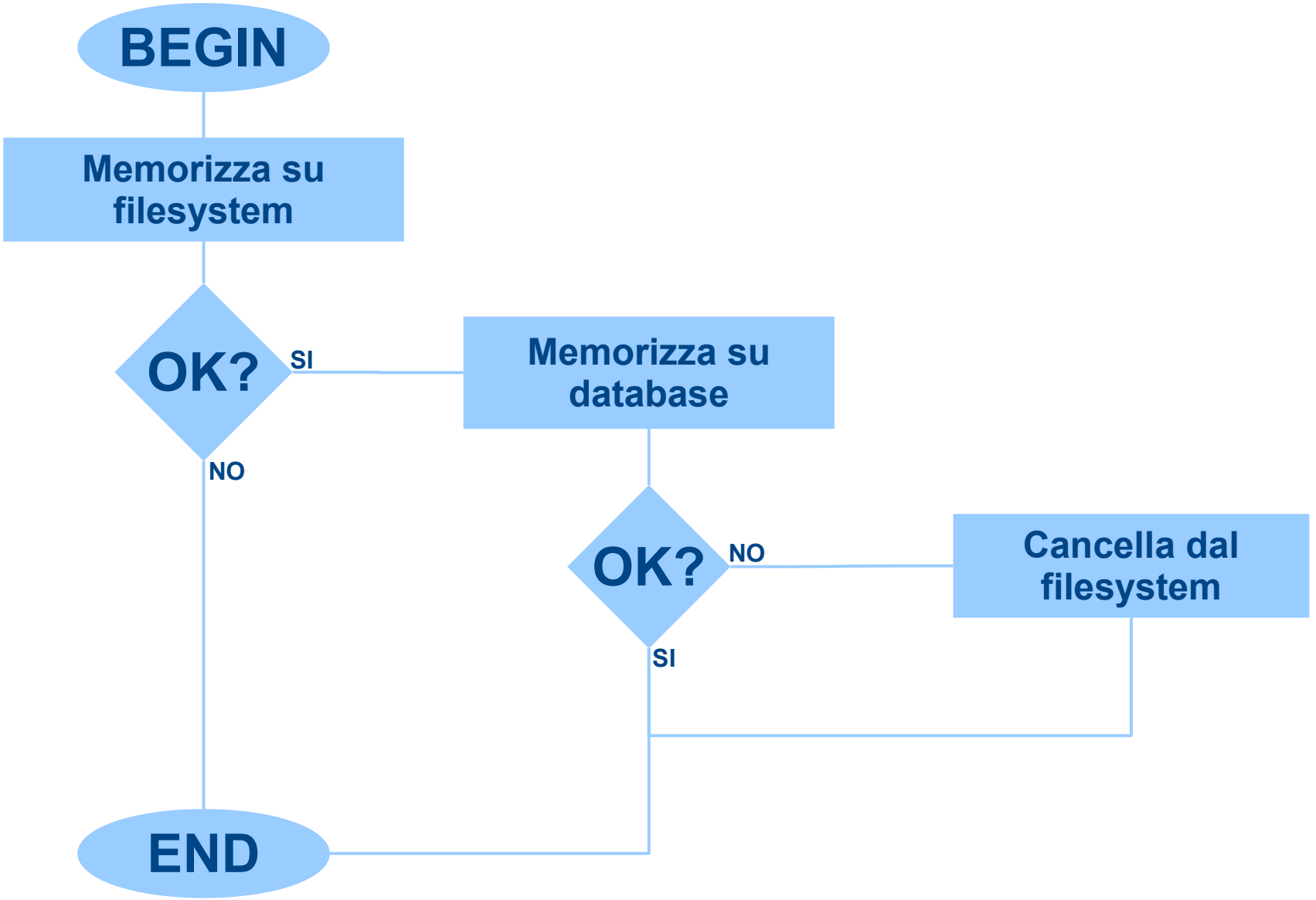
www.itpug.org
www.postgresql.org

Chi sono?

- Database administrator e developer in 2ndQuadrant Italia:
  - Database OLTP in ambienti business critical
  - Data warehousing
- Membro della comunità PostgreSQL e Linux
  - ITPUG
  - Linux Audio Italia
  - Nickname su Freenode: carloratm
    - #postgresql-it
    - #repmgr
    - #archlinux.it

Sommario

- Memorizzare dati binari nei database
- Memorizzare dati binari in PostgreSQL
  - LOB
  - Bytea
- Considerazioni finali
  - File system o database?


BEGIN
Memorizza su filesystem
OK?
SI
NO
Memorizza su database
OK?
NO
SI
Cancella dal filesystem
END


Transazione!
BEGIN
Memorizza su database
COMMIT

Hai veramente bisogno di memorizzare dati binari direttamente nel database?

Risposta comoda: NO

Risposta saggia: VEDIAMO
(probabilmente è sì)

Dati binari nel Database

- Cosa dichiara lo standard SQL
- Cosa mette a disposizione PostgreSQL
  - LOB
  - Bytea
- Solitamente dato binario = un file
  - Ma non necessariamente

Standard SQL:2008

- Stringa binaria
  - stringa di byte di dimensione fissa/variabile
- Stringa binaria *large object*
  - stringa di byte di dimensione variabile
  - normalmente di dimensione maggiore della binaria
- La dimensione massima è decisa dal DBMS

PostgreSQL e i dati binari

- 2 implementazioni differenti:
  - **LOB** (Large OBject)
  - **Bytea** (Byte Array)

LOB - Large OBject

- È un'**API**
  - Oggetti identificati con **`oid`**
  - Memorizzati in **`pg_largeobject`**
- Divisione in parti più piccole (*chunk*)
- Ogni chunk viene salvato in una riga
- Ogni LOB ha un proprietario e dei permessi associati
  - a partire da PostgreSQL 9.0
- Non usa TOAST

LOB

```
db=> CREATE TABLE tabella ( canzone oid );
CREATE TABLE

db=> \d pg_largeobject;
Table "pg_catalog.pg_largeobject"
 Column |   Type  | Modifiers
--------+---------+-----------
 loid   | oid     | not null
 pageno | integer | not null
 data   | bytea   |
Indexes:
    "pg_largeobject_loid_pn_index" UNIQUE, btree (loid, pageno)
```

LOB API

- Modellata sull'interfaccia Unix per la gestione dei file:
  - `open`
  - `read`
  - `write`
  - `seek`
- Lato client
  - psql, C, Java, PHP, Python, ecc.
- Lato server
  - SQL

Esempi

```
/* Importazione di un file come LOB */
Oid lo_import(PGconn *conn, const char *filename);

/* Esportazione di un LOB come file */
Int lo_export(PGconn *conn,
  Oid lobjId, const char *filename);

/* Scrivere dati in un LOB */
int lo_write(PGconn *conn,
  int fd, /* file descriptor aperto con lo_open */
  const char *buf, size_t n);
```

Vantaggi dei LOB

- Supporto file **fino a 2 GigaByte**
- Accesso random a parti del file
  - Stile “stream”

Svantaggi dei LOB

- Salvati in un'unica tabella **`pg_largeobject`**
- Manutenzione particolare
  - **`vacuumlo`**
- Utilizzo dell'API:
  - Maggiore complessità di sviluppo
  - Ulteriore passaggio per la memorizzazione

Bytea

- **È un tipo di dato**
  - Può essere usato come una colonna di una tabella
- Byte Array
  - Stringa di byte
- Dato atomico
  - Accesso solo sequenziale
- Usa TOAST
- Rappresentazione in 2 codifiche diverse:
  - Formato Hex
  - Formato Escape

Bytea - il formato Hex

- Hex = Esadecimale
- Ogni byte è codificato con 2 cifre esadecimali
- Introdotto in PostgreSQL 9.0
- L'intera stringa è preceduta dal prefisso **\x**
- Esempio:
  - `SELECT '\xC0FFEE';`

Bytea – il formato Escape

- Sequenza di caratteri ASCII
- Caratteri non rappresentabili in ASCII diventano speciali sequenze di escape
- Esempio:
  - `SELECT 'HELLO WORLD \007'::bytea;`

I vantaggi dei Bytea

- Salvati direttamente nella tabella
  - Come campo
- Manutenzione ordinaria
- Accesso più semplice e veloce
  - Non necessita di alcuna interfaccia
  - Supportato nativamente dai principali linguaggi di programmazione

Gli svantaggi dei Bytea

- Dimensione massima **1 GigaByte**
- Trattato come dato atomico
  - Ogni modifica richiede sostituzione in blocco del contenuto

Esempio di applicazione in C

- https://github.com/2ndquadrant-it/lob-bytea-demo
- GPL
- Versione Java disponibile
- Supporta sia LOB che bytea
- Vi permette di fare test e benchmark

Esempi di caricamento

```
/* Caricamento di un LOB */
./lotest "dbname=test" lob import /path/to/file

/* Caricamento di un Bytea */
./lotest "dbname=test" bytea import /path/to/file
```

Alcuni dati

		LOB	Bytea	+/-
10 file da 10 K	Tempo (secondi)	**0,103**	**0,099**	3.8%
	Spazio occupato	**160 kB**	**168 kB**	-5%
10 file da 100 K	Tempo (secondi)	**0,179**	**0,113**	36.8%
	Spazio occupato	**1360 kB**	**1104 kB**	18.8%
10 file da 1 M	Tempo (secondi)	**0,488**	**0,357**	26.8%
	Spazio occupato	**13 MB**	**10 MB**	23.0%
10 file da 10 M	Tempo (secondi)	**7,051**	**6,117**	13.2%
	Spazio occupato	**133 MB**	**104 MB**	21.8%
10 file da 50 M	Tempo (secondi)	**35,844**	**26,536**	25.9%
	Spazio occupato	**667 MB**	**519 MB**	22.1%

Come scegliere fra LOB o Bytea?

- Qual è la dimensione massima dei file?
- Devo poter modificare parti del file?
- Devo permettere l'accesso a parti del file?

Tabella riepilogativa

- Se file > 2GB = file system
- Se file > 1GB = LOB o file system
- Se file < 1 GB:
  - Se accesso random = LOB o file system
  - Altrimenti da valutare seriamente l'impiego di bytea

Conclusioni

- Memorizzare dati binari nel database
  - Opzione da non escludere a priori
  - Adatta per ambienti business critical
  - No manutenzione di documenti sul filesystem
- Consistenza e integrità dei dati (**transazioni!**)
- Sicurezza e protezione dei dati (GRANT)
- Alta disponibilità, backup e disaster recovery
- Scalabilità dei dati in sola lettura (hot standby)
- Distribuzione dei dati su più nodi (PL/Proxy)

Domande?

- Email: carlo.ascani@2ndquadrant.it
- www.repmgr.org

- wwww.2ndquadrant.it
- http://blog.2ndquadrant.it

Licenza Creative Commons

Attribuzione

Non commerciale

Condividi allo stesso modo

2.5 Italia

http://creativecommons.org/licenses/by-nc-sa/2.5/it/