

PARTITIONING



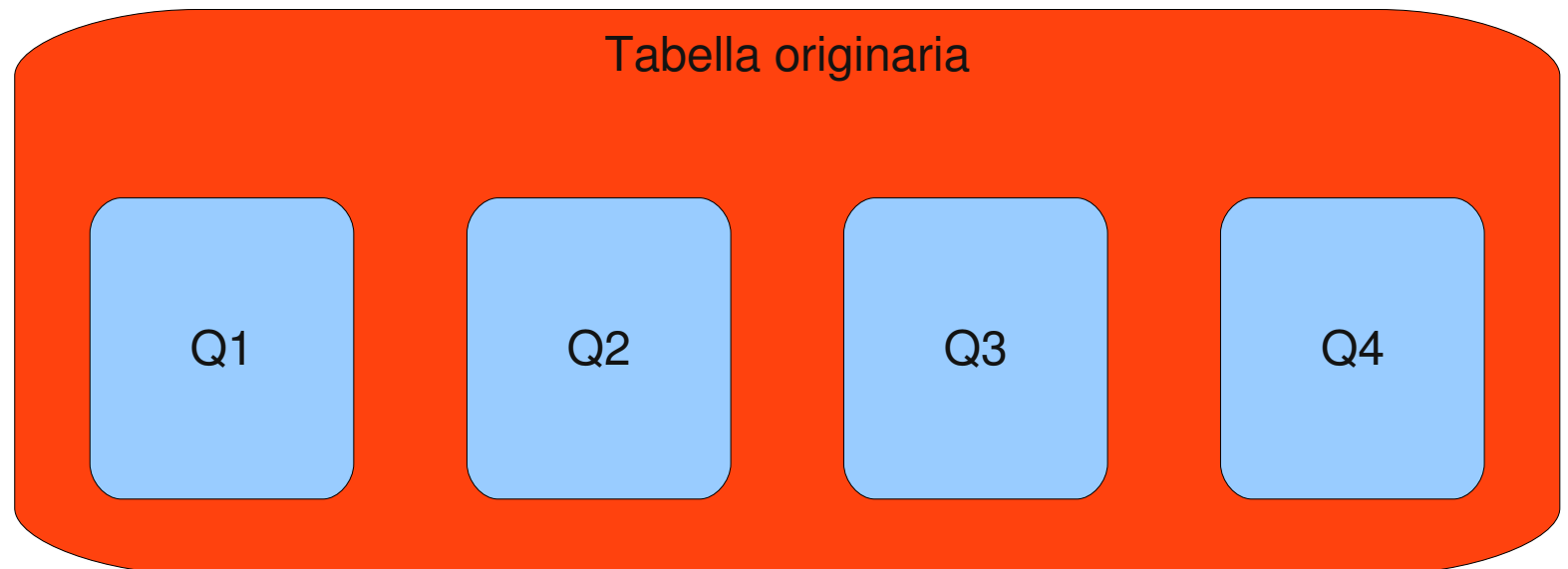
PARTITIONING

PARTITIONING

- Cos'è il partitioning
- Metodi per eseguire il partitioning
- Partition setup
- Partition table explain plain
- Tablespace

Cos'è il partitioning?

- Partitioning è dividere una tabella con tanti record in parti più piccole
- Benefici:
 - Aumento delle prestazioni di ricerca e aggiornamento
 - Maggiore facilità nel migrare i dati della tabella su supporti di storage più economici.



Overview

- Partitioning significa dividere in piccole tabelle il contenuto di una tabella di grande dimensione
- Le performance possono essere incrementate drasticamente per certi tipi di ricerche. La performance di update vengono incrementate.
- Gli indici possono entrare in memoria perchè di più piccola dimensione
- Alle volte le operazioni di delete possono equivalere semplicemente con il rimuovere una delle partizioni
- I dati possono essere passati su supporti più piccoli quindi più economici e/o più veloci
- PostgreSQL realizza il partitioning attraverso l'ereditarietà delle tabelle. Ogni partizione deve essere creata come una tabella figlia di una singola tabella madre. La tabella madre esiste solo per rappresentare il dataset ed è normalmente vuota.

Metodi per il partitioning

- Range Partitioning
 - Le partizioni sono definite attraverso dei vincoli sulla chiave primaria e non devono essere presenti sovrapposizioni o buchi.

- List Partitioning
 - Ogni valore della chiave primaria viene specificato in una lista

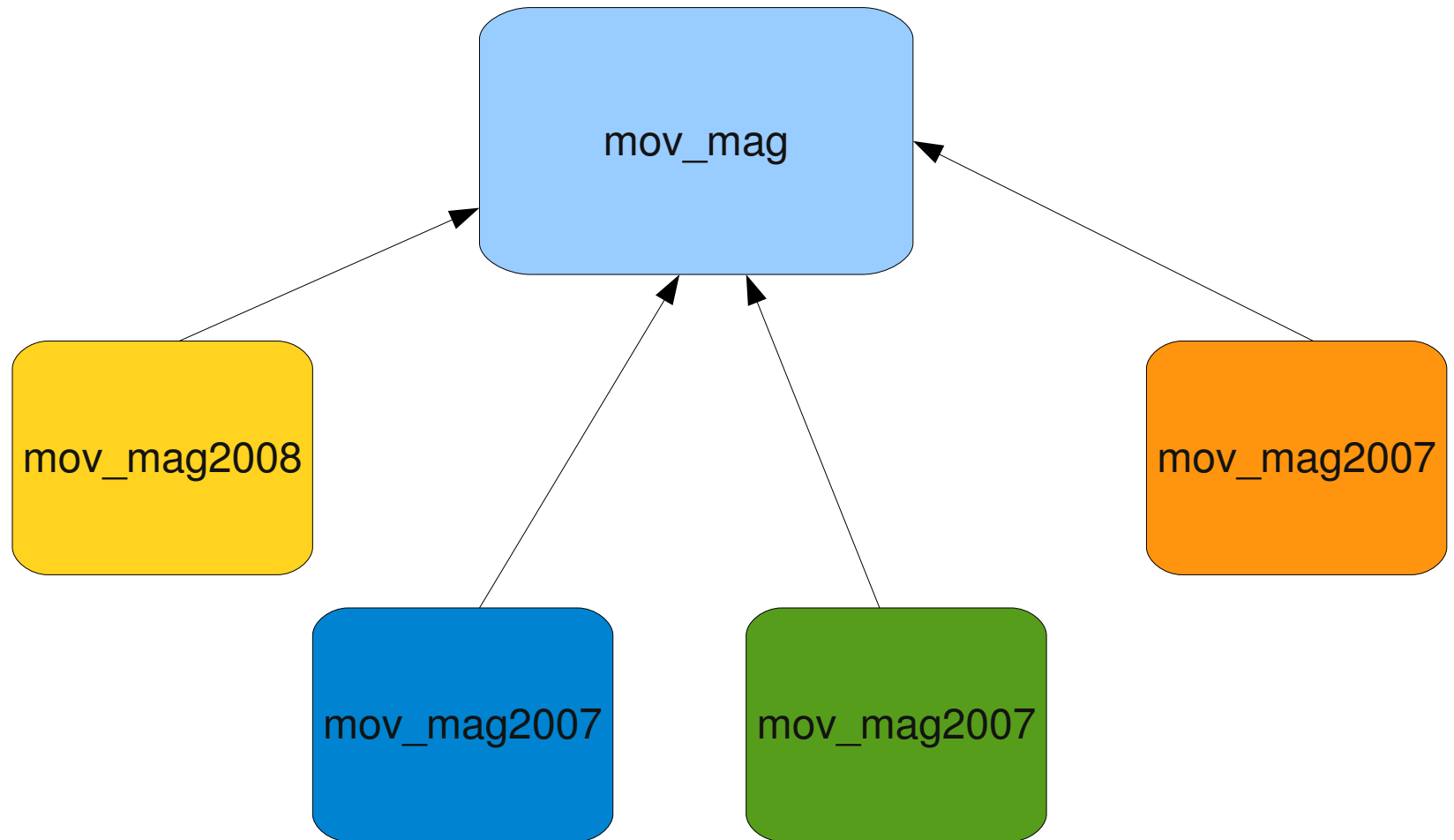
Partitioning Setup

- Creare la master table (tabella madre)
 - Questa tabella non conterrà i dati, definirà solamente i vincoli che verranno applicati a tutte le partizioni.
- Creare le tabelle figlie attraverso l'ereditarietà con la tabella master
- Aggiungere i vincoli nella tabella figlia per definire i vincoli attraverso i quali i valori chiave vengono accettati
 - Alcuni esempi: `check (x = 1)`
 - `Check (provincia in ('MI','TO','BO'))`
 - `Check (id > 100 and id < 200)`
- Verificare che non ci siano sovrapposizioni di valori nelle tabelle figlie

Partitioning Setup

- Creare gli indici necessari in ogni tabella figlia
- Definire le regole o i trigger per ridirigere i dati dalla tabella master verso la tabella figlia appropriata.
- Assicurarsi che `constraint_exclusion` in `postgresql.conf` sia impostato su `true`

Esempio



Partitioned table explain plan

```
explain SELECT * from mov_mag where campo1 = 'val1' and campo2 = 'val2';
```

QUERY PLAN

Result (cost=0.00..29.83 rows=2 width=80)

-> Append (cost=0.00..29.83 rows=2 width=80)

-> Seq Scan on mov_mag (cost=0.00..21.55 rows=1 width=76)

Filter: ((campo1 = 'val1'::bpchar) AND ((campo2)::text = 'val2'::text))

-> Index Scan using citidx1 on mov_mag_2008 mov_mag (cost=0.00..8.28 rows=1 width=80)

Index Cond: ((campo2)::text = 'val2'::text)

Filter: ("campo1" = 'val1'::bpchar)

(7 rows)

Creazione Master Table

```
CREATE TABLE vendite (  
    id INTEGER NOT NULL,  
    data_vendita DATE,  
    somma NUMERIC(9,2)  
);
```

Creazione delle tabelle figlie

```
CREATE TABLE vendite_08q1 (  
    CHECK (data_vendita >= DATE '2008-01-01' AND  
        data_vendita < DATE '2008-04-01' )  
)  
INHERITS (vendite);
```

```
CREATE TABLE vendite_08q2 (  
    CHECK (data_vendita >= DATE '2008-04-01' AND  
        data_vendita < DATE '2008-07-01' )  
)  
INHERITS (vendite);
```

Creazione delle tabelle figlie

```
CREATE TABLE vendite_08q3 (  
    CHECK (data_vendita >= DATE '2008-07-01' AND  
           data_vendita < DATE '2008-10-01' )  
)  
INHERITS (vendite);
```

```
CREATE TABLE vendite_08q4 (  
    CHECK (data_vendita >= DATE '2008-10-01' AND  
           data_vendita < DATE '2009-01-01' )  
)  
INHERITS (vendite);
```

Creazione degli indici

- `CREATE INDEX vendite_08q1_data_vendita on vendite_08q1 (data_vendita);`
- `CREATE INDEX vendite_08q2_data_vendita on vendite_08q2 (data_vendita);`
- `CREATE INDEX vendite_08q3_data_vendita on vendite_08q3 (data_vendita);`
- `CREATE INDEX vendite_08q4_data_vendita on vendite_08q4 (data_vendita);`

Creazione delle RULES

```
CREATE RULE inserimento_vendita_08q1 AS
ON INSERT TO vendite WHERE
(data_vendita >= DATE '2008-01-01' AND
 data_vendita < DATE '2008-04-01')
DO INSTEAD
INSERT INTO vendite_08q1(NEW.
 id,NEW.data_vendita,NEW.somma);
```

STESSA COSA PER LE ALTRE TABELLE FIGLIE

CREAZIONE DEL TRIGGER

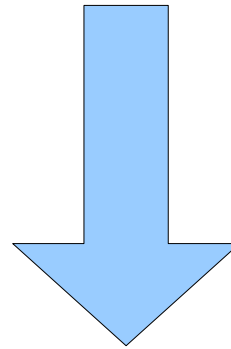
```
CREATE OR REPLACE function f_trigger() RETURNS
trigger
as $$
begin
IF
    NEW.data_vendita >= DATE '2008-01-01'
    AND NEW.data_vendita < DATE '2008-04-01' THEN
    INSERT INTO vendite_08q1 values
    (NEW.id,NEW.data_vendita,NEW.somma);
ELSIF
```

CREAZIONE DEL TRIGGER

```
NEW.data_vendita >= DATE '2008-04-01'  
AND NEW.data_vendita < DATE '2008-07-01' THEN  
  
    INSERT INTO vendite_08q2 values  
    (NEW.id,NEW.data_vendita,NEW.somma);  
  
ELSIF  
  
    .....  
  
END IF;  
  
RETURN NULL;  
  
END;  
  
$$  
  
language plpgsql;
```

CREAZIONE DEL TRIGGER

- CREATE TRIGGER miottrigger BEFORE INSERT ON vendite FOR EACH ROW EXECUTE PROCEDURE f_trigger();



A questo punto la procedura di partitioning è terminata

Partitioning : manutenzione

- Esistono diversi modi per rimuovere una partizione
 - `DROP TABLE nome_partizione`
 - E' preferibile di più però `ALTER TABLE nome_partizione_NO INHERIT master_table`

- Se invece vogliamo cancellare la `master_table` e tutte le sue partizioni :
 - `DROP TABLE master_table CASCADE`

CAVEATS

- Non c'è un metodo per verificare che tutti i vincoli imposti siano mutuamente esclusivi
- Non c'è un metodo per specificare che nessuna riga deve essere inserita nella master_table → senza TRIGGER ovviamente
- Dal momento che la master_table non contiene dati non è possibile assegnare alcuna foreign key alla master_table

PARTITIONING



- Cos'è il partitioning
- Metodi per eseguire il partitioning
- Partition setup
- Partition table explain plain
- Tablespace

