**PARTE 3.9: PROGETTO FISICO (su socialmarketshort)**

**Progettazione fisica** (carico di lavoro basato sullo schema sottostante)

SET search\_path TO 'socialmarketshort';

**PRIMA QUERY**

/\*

Prima query del workload scelto, facciamo semplicemente una select dell’id, del nome e della tipologia di tutti i prodotti nell’inventario

\*/

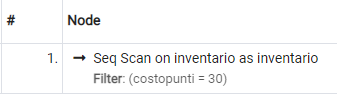
SELECT idProdotto, nomeProdotto, tipologiaProdotto

FROM INVENTARIO

WHERE costopunti = 30;

**PRIMA DELLA CREAZIONE DELL’INDICE**





**Successfully run. Total query runtime: 170 msec.**

**CREAZIONE DELL’INDICE**

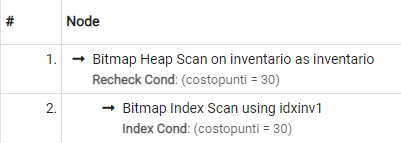
CREATE INDEX idxInv1 ON INVENTARIO(costopunti);

CLUSTER INVENTARIO USING idxInv1;

-- DROP INDEX idxInv1;

**DOPO LA CREAZIONE DELL’INDICE**





**Successfully run. Total query runtime: 122 msec.**

**COMMENTO**

Prima della creazione dell’indice viene scelto dal sistema un piano di esecuzione di tipo SEQ SCAN (scannerizzazione sequenziale senza utilizzo di indici) applicando un filtro sulla condizione di selezione costoPunti = 30. Si sceglie un indice ordinato rispetto ad un hash dato che non si può prevedere la presenza di aree di overflow e data l’implementazione non consigliata su PostgreSQL. Una volta applicato l’indice idxInv1 sul campo costoPunti, data la bassa probabilità che una tupla di INVENTARIO soddisfi la condizione (fattore di selettività), il piano d’esecuzione considerato varia. Con un predicato del tipo “costoPunti > 30” si osserva infatti il mantenimento del piano SEQ SCAN, dato il fattore di selettività più alto.

Il piano d’esecuzione con indice risulta essere un IDX SCAN con condizione costoPunti = 30, che permette una riduzione, anche se lieve, del tempo di esecuzione. Dato il comunque non troppo alto numero di tuple contenute nella relazione si sono svolti più tentativi per evitare di prendere erroneamente un caso decisamente peggiore nell’IDX scan e uno migliore nel SEQ SCAN.

Anche se con tale numero di tuple non si riescono ad osservare grandi variazioni rispetto alla versione non clusterizzata, si è pensato di clusterizzare INVENTARIO sulla base dell’indice su costoPunti, pensando che le condizioni su questo attributo possano essere più comuni di condizioni su qtScaricato.

**SECONDA QUERY**

/\*

Seconda query delle tre a scelta. Si vanno a

cercare i prodotti nell'inventario (selezionandone id, nome e quantità presente) che soddisfano una determinata condizione complessa.

Nello specifico la condizione richiede che o la quantità sia 10 o il costoPunti sia maggiore di 2, che l’id del prodotto sia diverso da 1 e che non sia mai stato scaricato.

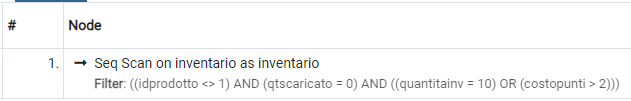
\*/

SELECT idProdotto, nomeProdotto, quantitaInv

FROM INVENTARIO

WHERE (quantitaInv = 10 OR costoPunti > 2) AND idProdotto <> 1 AND qtScaricato = 0;

**PRIMA DELLA CREAZIONE DELL’INDICE**



**Successfully run. Total query runtime: 212 msec.**

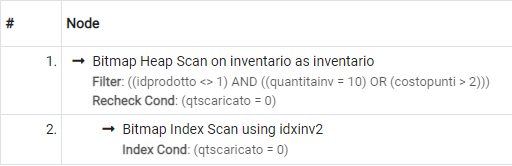
**CREAZIONE DELL’INDICE**

CREATE INDEX idxInv2 ON INVENTARIO(qtScaricato);

--DROP INDEX idxInv2;

**DOPO LA CREAZIONE DELL’INDICE**

****

****

**Successfully run. Total query runtime: 174 msec.**

**COMMENTO**

Anche per questa seconda Query con condizione complessa il piano considerato previa creazione dell’indice è il SEQ SCAN, mentre successivamente risulta essere l’IDX SCAN su qtScaricato. Viene scelta la creazione dell’indice per l’attributo qtScaricato poiché la scelta del cammino d’accesso da parte del sistema ricadrebbe comunque su di esso. Nel predicato scelto, infatti, “qtScaricato = 0” risulta essere il fattore booleano, ossia una condizione che, se falsa, rende falsa tutta l’interrogazione. Come per prima si registra un miglioramento medio delle prestazioni della query con cammino d’accesso con indice.

**TERZA QUERY**

/\*

Terza ed ultima query di quelle a scelta, in questo caso la query comprende un join dove si richiede di selezionare i volontari con i rispettivi turni

(limitandosi ai volontati per cui è registrato almeno un turno. Si seleziona codice fiscale, cognome e nome dei volontari,

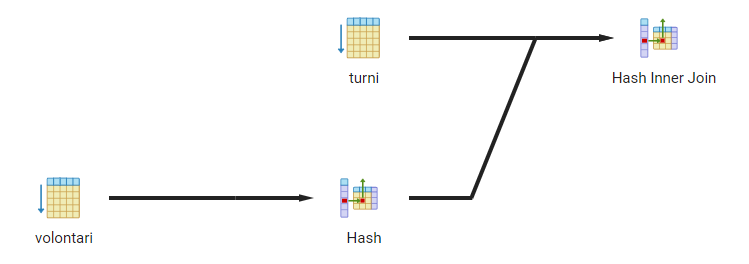
insieme alla data e all’orario di inizio e di fine dei loro rispettivi turni.

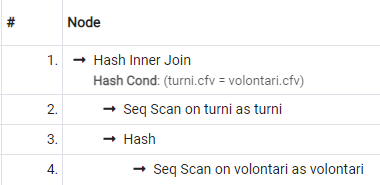
\*/

SELECT cfV, cognomeV, nomeV, dataTurno, orarioInizio, orarioFine

FROM VOLONTARI NATURAL JOIN TURNI

**PRIMA DELLA CREAZIONE DELL’INDICE = DOPO LA CREAZIONE DELL’INDICE**





**Successfully run. Total query runtime: 188 msec.**

**CREAZIONE DELL’INDICE**

-- CREATE INDEX idxTurni ON TURNI(cfV);

-- DROP INDEX idxTurni;

**COMMENTO**

Nel caso della terza query, senza disabilitare l’hash join con il comando “SET enable\_hashjoin=off;” viene considerato il piano di Hash Inner Join anche in presenza dell’indice scelto (quello sul cfV presente in TURNI). Disabilitando l’hash join viene scelto, in presenza dell’indice, un piano di tipo Inner Merge Join con SORT precedente su Volontari. Non risultando alcun miglioramento sul tempo di esecuzione si sceglie per il piano fisico di non creare alcun indice per questa interrogazione.

NB: Anche utilizzando tutti i seguenti indici, al fine di permettere il Merge Join, viene comunque considerato l’Hash Join:

CREATE INDEX idxVol ON VOLONTARI(cfV);

CREATE INDEX idxTurni ON TURNI(cfV);

CLUSTER VOLONTARI USING idxVol;

CLUSTER TURNI USING idxTurni;

**INDICI SCELTI**

CREATE INDEX idxInv1 ON INVENTARIO(costopunti);

CLUSTER INVENTARIO USING idxInv1;

CREATE INDEX idxInv2 ON INVENTARIO(qtScaricato);

***------------- Tabella tuple inserite -------------***

| ***TABELLA*** | ***TUPLE INSERITE*** | ***BLOCCHI*** |
| --- | --- | --- |
| *INVENTARIO* | *4000* | *90* |
| *VOLONTARI* | *4000* | *101* |
| *TURNI* | *4000* | *87* |

***------------- Query utili sui cataloghi -------------***

--Query sugli oggetti del DB vista a laboratorio

SELECT C.oid, relname, relfilenode, relam, relpages, reltuples, relhasindex, relkind

FROM pg\_namespace N JOIN pg\_class C ON N.oid = C.relnamespace

WHERE N.nspname = 'socialmarketshort';

--Ritorna gli indici di socialmarketshort

SELECT C.oid, relname, relam, relpages, relkind, indexrelid, indrelid, indnatts, indisunique, indisprimary, indisclustered, indkey

FROM (pg\_namespace N JOIN pg\_class C ON N.oid = C.relnamespace) JOIN pg\_index ON C.oid = indexrelid

WHERE N.nspname = 'socialmarketshort';

--data size per le tabelle dello schema socialmarketshort

SELECT schemaname as table\_schema, relname as table\_name, pg\_size\_pretty(pg\_relation\_size(relid)) as data\_size

FROM pg\_catalog.pg\_statio\_user\_tables

WHERE schemaname = 'socialmarketshort';

**PARTE 3.10: TRANSAZIONE**

**SCRIPT TRANSAZIONE (T1):**

set search\_path to 'socialmarket';

SET datestyle to 'MDY';

-- Transazione basata sui dati presenti nelle insert manuali del file

-- Duplico l'importo donato della donazione del '03-20-2021' (l’ultima) -- … effettuata dal donatore con mail: donatore1@gmail.com

BEGIN;

SET CONSTRAINTS ALL IMMEDIATE;

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

SELECT telD

FROM DONATORI

WHERE mail = 'donatore1@gmail.com';

-- ipotesi di aver trovato questo telD

SELECT idD

FROM DONAZIONI

WHERE telD = '3456664510'

AND dataD = '03-20-2021';

-- ipotesi di aver trovato questo idD

UPDATE DONAZIONI SET Importo = Importo + Importo \* 2 WHERE idD = 50000;

SELECT \*

FROM DONAZIONI

WHERE idD = 50000;

COMMIT;

**LIVELLO d’ISOLAMENTO:**

| Livello di isolamento | Lost update | Dirty Read | Unrep. Read | Phantom Row |
| --- | --- | --- | --- | --- |
| REPEATABLE READ | NO | NO | NO | YES |

Si sceglie REPEATABLE READ (protocollo di locking Strong 2 Phase Locking) come livello di isolamento poiché non si vuole permettere il verificarsi delle anomalie di Lost Update, Dirty Read e Unrepeatable Read. Si vuole che nessun valore letto o scritto da T1 venga modificato da altre transazioni prima che T1 abbia terminato.

L’anomalia di Phantom Row non arreca problemi alla transazione (non cambia il set di tuple ritornato) e per questo non si considera il livello SERIALIZABLE.

Si evita che, dopo ad esempio la lettura da parte di T1 del telefono associato alla mail, una transazione T2 operi una modifica su tale correlazione invalidando le successive operazioni sull’importo. Una successiva lettura da DONATORI evidenzierebbe il problema verificatosi sulla correlazione telD-mail:

SELECT telD FROM DONATORI WHERE mail = 'donatore1@gmail.com';

Si evita inoltre che, dopo la scrittura di T1, una transazione T2 legga l’importo modificato, invalidato dall’eventuale ROLLBACK poi effettuato da T1.

Nota: tutti i livelli di isolamento garantiscono il non verificarsi dell’anomalia di Lost Update

**PARTE 3.11: CONTROLLO DELL’ACCESSO**

**CONTROLLO DELL’ACCESSO**

| **TABELLE** | **Alice (gestore Social Market)** | **Roberto (volontario)** |
| --- | --- | --- |
| Appuntamenti | ALL | SELECT,UPDATE,INSERT |
| Enti | SELECT,UPDATE,INSERT | SELECT |
| Clienti | SELECT,UPDATE,INSERT | SELECT |
| Famigliari | SELECT,UPDATE,INSERT | SELECT |
| Merci | SELECT,UPDATE,INSERT | SELECT,UPDATE,INSERT |
| Inventario | SELECT,UPDATE,INSERT | SELECT,UPDATE,INSERT |
| Donazioni | ALL | SELECT,INSERT |
| Donatori | ALL | SELECT,INSERT |
| Volontari | ALL | X |
| Trasporti | SELECT,UPDATE,INSERT | SELECT,UPDATE |
| Turni | ALL | SELECT |
| Appartenenze | ALL | SELECT |
| Acquisti | SELECT, INSERT | SELECT, INSERT |

*ALL = SELECT,UPDATE,INSERT,DELETE (si ignorano TRUNCATE, TRIGGER, …)*

**Comandi e specifica permessi forniti:**

set search\_path to 'socialmarket';

CREATE USER alice PASSWORD ‘alice’;

GRANT USAGE ON SCHEMA socialmarket TO alice;

CREATE USER roberto PASSWORD ‘roberto’;

GRANT USAGE ON SCHEMA socialmarket TO roberto;

GRANT SELECT,INSERT,UPDATE ON ALL TABLES IN SCHEMA socialmarket TO alice;

REVOKE UPDATE ON ACQUISTI FROM alice CASCADE;

GRANT DELETE ON APPUNTAMENTI, DONAZIONI, DONATORI, VOLONTARI, TURNI, APPARTENENZE TO alice;

GRANT SELECT ON ALL TABLES IN SCHEMA socialmarket TO roberto;

REVOKE SELECT ON VOLONTARI FROM roberto CASCADE;

GRANT UPDATE,INSERT ON APPUNTAMENTI, MERCI, INVENTARIO TO roberto;

GRANT UPDATE ON TRASPORTI TO roberto;

GRANT INSERT ON DONAZIONI, DONATORI, ACQUISTI TO roberto;

--SET role alice;

–-SET role roberto;

SELECT table\_name, privilege\_type

FROM information\_schema.role\_table\_grants

where grantee = 'alice'

SELECT table\_name, privilege\_type

FROM information\_schema.role\_table\_grants

where grantee = ‘roberto’

ALICE:

In quanto parte dei gestori del Social Market stesso vengono forniti più privilegi rispetto ai volontari.

Si sceglie di dare tutti i privilegi di SELECT,INSERT,UPDATE (eccetto in ACQUISTI) in quanto, nonostante non si pensino i gestori del SocialMarket come DBA, si considera che essi abbiano il massimo potere decisionale sugli tutti gli aspetti di una normale gestione del database. Si pensa dunque che un gestore del Social Market debba essere in grado di visionare tutti i dati presenti nelle tabelle e di inserirne di nuovi, al fine di gestire correttamente le donazioni e gli acquisti dei prodotti, gli scarichi, gli appuntamenti, i turni e i trasporti effettuati dai volontari e, in generale, i dati (anche perchè non sensibili) di tutte le persone fisiche coinvolte nella base di dati (clienti titolari e nucleo famigliare associato, volontari e donatori).

In ACQUISTI si decide di revocare:

* L’Aggiornamento, in quanto in situazioni normali si pensa come una tabella di soli inserimenti. Un’eccezionale update di un acquisto, che nello svolgimento abituale delle mansioni non viene previsto, si delega al DBA.

Inoltre:

* Si fornsice il privilegio di cancellazione a APPUNTAMENTI, DONAZIONI, DONATORI, VOLONTARI, TURNI, APPARTENENZE. Nel caso delle prime quattro tabelle la DELETE risulta ovviamente limitata ai soli casi in cui tale cancellazione non provoca effetti a cascata nella base di dati, laddove vengano toccate tabelle che non possono subire cancellazioni. Ad esempio si pensa ragionevole poter cancellare un donatore se entro un intervallo di tempo non ha ancora effettuato donazioni, e similmente con altre tabelle. Si è considerata la cancellazione come un privilegio dunque meno necessario e più forte, in quanto si ritiene meno frequente e legato a situazioni eccezionali.
* Il privilegio di cancellazione non viene dato su ENTI perchè si pensa che si voglia mantenere traccia dei rapporti che un azienda ha avuto in passato con il Market, salvo situazioni eccezionali gestite da utenti di ruolo maggiore come il DBA.
* Il privilegio di cancellazione non viene dato su CLIENTI, FAMIGLIARI, TRASOPORTI, MERCI e INVENTARIO perchè si pensa essere un pivilegio del solo DBA e dati gli effetti a cascata che può provocare nella base di dati.

ROBERTO:

Si decide di fornire a roberto:

* Il privilegio di selezione su tutte le tabelle eccetto Volontari, in modo tale che possa visualizzare i dati relativi ai prodotti, ai clienti e alla famiglia ad essi correlata e alle donazioni effettuate dai donatori, di cui non sono memorizzati dati sensibili. Si sceglie di non fornire la selezione su Volontari per isolare i dati di ogni volontario, visibili da ruoli superiori quali i gestori del Social Market.

Sono i ruoli superiori a inserire le appartenenze di un volontario a una specifica assocazione.

* I privilegi di aggiornamento e inserimento sugli APPUNTAMENTI, che il volontario deve poter gestire al meglio in quanto potenziale staff di accoglienza
* Il privilegio di inserimento su DONAZIONI e DONATORI, in modo tale che un Volontario possa inserire nel DB una nuova donazione sopraggiunta al Social Market, magari da un Donatore nuovo e dunque non presente nella base di dati.
* I privilegi aggiuntivi di aggiornamento e inserimento su MERCI e INVENTARIO, in modo tale che il volontario possa gestire correttamente i prodotti del Social Market. Ad esempio in INVENTARIO potrebbe essere necessario l’aggiornamento della quantità di prodotti scaricati, nei momenti in cui i volontari raccolgono i prodotti nei cestelli. Si delega al DBA l’eventuale eccezionale cancellazione da tali tabelle.
* Il privilegio aggiuntivo di aggiornamento sui TRASPORTI, in modo tale da permettere al volontario la variazione degli scatoloni da portare o di altre informazioni riguardo il trasporto, quali ritardi nella consegna.
* Il privilegio aggiuntivo di inserimento sugli ACQUISTI, in modo tale che il volontario possa aggiungere i nuovi acqusiti fatti dai clienti da lui seguiti durante gli appuntamenti.