## I moderni Ruoli di Amministratore di DataBase in un ambiente Cloud

*"Gli Architetti disegnano progetti dettagliati prima che un solo mattone venga posizionato o un chiodo venga piantato, ma pochi programmatori delineano anche solo uno schizzo di che cosa fara' il loro programma prima di partire a scrivere direttamente il codice finale"*

**– Leslie Lamport**

*"Trovo che risparmiare qualche ora di pianificazione mi fa spendere settimane di scrittura del codice ed esecuzione dei test."*

– Anonimo

***Sommario***



# Introduzione - I database nei moderni IT Stacks

Qualche forma di database è sempre esistita all'interno del mondo IT fin dai primi anni 70, quando IBM creò il System R ed il linguaggio SQL.

In quei primi tempi l'hardware era costoso mentre il personale impiegatizio era relativamente economico, quindi gli Amministratori di Database (DBA) incaricati lavoravano con gli sviluppatori per ottenere il massimo dall'hardware che avevano a disposizione.

Non appena l'hardware ed il software sono calati di prezzo e nel frattempo sono anche diventati più potenti, uno dei maggiori affari della nostra attuale era tecnologica, il costo del personale è diventato la più grande componente delle maggiori organizzazioni IT.

Tuttavia la “commoditizzazione” dell’hardware ha un aspetto negativo, nel fatto che le aziende hanno dovuto gestire un rapido deprezzamento delle infrastutture IT, con un ciclo di vita tipico di 36 mesi per l’hardware sia dal punto di vista tecnologico che contabile. Il leasing emerse come una soluzione parziale a questo problema di rapida obsolescenza, ma era uno strumento imperfetto.

### *Cloud Computing (Avere dei computer in Cloud)*

Per risolvere questi problemi è emerso il Cloud Computing come tendenza del settore informatico, sotto forma di esternalizzazione verso terze parti sia dell’hardware che del relativo personale, come amministratori di sistema e amministratori di Database (DBA). Questi venditori specializzati in Cloud sono in grado di operare in modo efficiente grazie all’economia di scala e ottengono un maggior utilizzo dell’hardware grazie al fatto che supportano più Clienti su server condivisi e virtualizzati.

All’inizio questi servizi erano delle novità benvenute, specialmente per le piccole organizzazioni che erano in grado di accedere ad un’ampia tipologia di strutture all’interno dell’ecosistema del Cloud, il tutto ad un costo inizialmente modesto. Accadde lo stesso fenomeno che abbiamo visto con Uber in contrapposizione alle compagnie tradizionali di taxi, i venditori dei servizi Cloud, in competizione tra di loro, hanno abbassato i prezzi ad un livello interessante.

Ma saltare sul carro di questo nuovo mondo ha generato dei problemi, che solo adesso stanno diventando chiari.

E’ fuor di dubbio che i venditori di Cloud ti concedano in affitto l’accesso ad un database di tua scelta funzionante in uno dei loro centro dati.

Oppure, per coloro che necessitano di un maggior controllo dell’ambiente informatico, i venditori possono fornire, ad un costo maggiore, server virtuali o dedicati dove è possibile usare comandi “ssh” per connettersi a una struttura remota e per installare il proprio database tramite la tradizionale riga di comando.

### *Problema della mancanza di un DBA di produzione*

Per gestire i livelli più bassi dello “stack IT”, i venditori di servizi cloud fanno più o meno bene ad erogare alcuni dei servizi che un’organizzazione richiede ad un DBA (Amministratore di Database) di produzione.

Ma quello che questi venditori non possono fare correttamente è prendere delle buone decisioni a livello alto nell’architettura di sistema. Spostare i server fuori e licenziare l’Amministratore di Database (DBA), non elimina la necessità di servizi di progettazione del DBA, specialmente quando il database dell’applicazione aumenta di dimensione.

La mancanza del DBA spesso non si nota all’inizio, quando le quantità di dati sono modeste; ma nel tempo si profilano delle difficoltà nella gestione di grande mole di dati, dovute a decisioni non ottimali prese durante lo sviluppo e la realizzazione del database.

La reazione tipica delle aziende in questa situazione è credere che i loro problemi possano essere risolti con una partizione Cloud più grande. Ne deriva quindi un processo costoso in cui le organizzazioni esplorano le varie opzioni di aumento delle performance che i venditori di Cloud possono offrire.

Questo approccio può far guadagnare un po' di tempo, ma in realtà cura soltanto i sintomi di un altro vero problema – una probabile progettazione non ottimale.

La motivazione sottostante di questo problema è che togliere dalla responsabilità del team il DB di produzione non elimina la necessità delle competenze di un DBA.

***Problema della mancanza di un DBA di sviluppo***

Parte della motivazione per muoversi verso un ambiente Cloud era dovuta al fatto che gli sviluppatori erano in grado di estendere il loro ruolo e coprire il lavoro dell'Amministratore di Database, in aggiunta al loro consueto lavoro finalizzato all'applicazione.

Questo è' in parte vero, poiché le competenze dello sviluppatore e dell'Amministratore di Database (DBA) si sovrappongono nell'area dello stack relativo all'applicazione.

Tradizionalmente gli Amministratori di Database (DBA) assistevano i programmatori durante lo sviluppo e fornivano consigli via via che il progetto avanzava. I DBA creavano strutture di database corrette e performanti e lavoravano a risolvere i problemi di scarsa performance che gli sviluppatori potevano incontrare.

Questo spiega perché gli Amministratori di Database avevano un ruolo diverso dagli sviluppatori. Anche se gli sviluppatori spesso conoscono le basi per lavorare con database SQL, di solito essi hanno carenza di formazione e di interesse nel conoscerli approfonditamente; di conseguenza nascono dei problemi. Nello specifico ne derivano delle scelte non ottimali prese a causa di competenze insufficienti.

Ci si può chiedere perché gli sviluppatori tipicamente siano carenti di queste abilità e ci sono diversi motivi:

1. Al giorno d'oggi le scuole minimizzano il linguaggio SQL ed i database relazionali per mancanza di spazio nel programma scolastico.

2. L'hardware è divenuto più veloce, così alcuni dei problemi di performance vengono mascherati da un'esecuzione più veloce e da CPU aggiuntive.

3. I database SQL sono visti come una "tecnologia vecchia" e si crede che una tecnologia più nuova sia anche migliore.

4. Lo sviluppo personale finalizzato al proprio curriculum vitae è un fattore significativo all'interno delle aziende. Questo significa che gli sviluppatori desiderano fortemente utilizzare le tecnologie più recenti, in modo da avere successivamente sui loro CV qualcosa di rivendibile.

Proseguendo in questo modo dove ci porta tutto questo? Perché è così' negativo?

Uno specifico comportamento problematico che riscontriamo ripetutamente è che gli sviluppatori, non trovandosi a proprio agio con le tabelle dove solitamente il database SQL piazza i suoi dati, mettono tutti i dati per la loro applicazione in un'unica grande tabella. In pratica gli sviluppatori convertono un sofisticato database con le sue avanzate capacità di elaborazione in un singolo foglio di calcolo, al quale accedono attraverso la loro applicazione.

Il meglio che si possa dire di questo approccio è che risulta semplice, ma priva l'organizzazione che consente questo tipo di decisione progettuale di molti benefici che si otterrebbero facilmente con scelte più intelligenti.

Come abbiamo già visto in precedenza rimuovere dal team l'Amministratore di Database (DBA) di sviluppo non toglie la necessità delle competenze di un DBA.

### *Il graduale incremento nell'utilizzo del Cloud*

Parlando ai clienti di servizi Cloud, ci è balzato agli occhi questo altro aspetto: il graduale incremento nell'utilizzo e quindi nel costo dei servizi acquistati.

Parte di questa crescita del costo deriva da quanto menzionato in precedenza: un improvvisato DBA ("Graphical User Interface DBA") che cerca di risolvere problemi di architettura del sistema tramite l'acquisto di vari pacchetti aggiuntivi, pubblicizzati nel menu dei servizi Cloud.

Un altro aspetto importante è l'effetto "lontano dagli occhi, lontano dai pensieri" nell'utilizzo del Cloud. Diciamo questo perché le fatture del servizio si presentano ogni mese e quindi la proliferazione di contrattempi di sviluppo, di test e di produzione non sono per niente evidenti, ed i relativi costi possono davvero crescere a dismisura.

Questi tentativi di "aggiustamento" della performance sono definitivamente molto costosi, ma non sempre efficaci poiché, senza avere correttamente diagnosticato i problemi sottostanti, è difficile prescrivere la giusta soluzione.

Ancora una volta diventa evidente che rimuovere dal team l'Amministratore di Database (DBA) non toglie la necessità delle competenze di un DBA.

# Chapter 1 - Amministratore di Database (DBA) di sviluppo del PostgreSQL

Questo ruolo è il più "social" e il più in alto nello stack di sviluppo. Il DBA di sviluppo assiste gli sviluppatori di applicazioni nel convertire i requisiti di business in un codice che funzioni, il quale utilizzi con efficienza le risorse del database. Nel fare questo l'azienda ottiene il massimo rendimento degli investimenti fatti sul suo database.

In aggiunta così l'azienda posticipa o evita completamente la necessità di investire su un hardware più veloce, risparmiando così tempo, soldi e riducendo il rischio operativo dell'organizzazione.

***Classe sui fondamentali del database relazionale***

Per riempire alcune lacune nell'educazione degli sviluppatori, abbiamo creato una classe che fornisce le basi su perché l'SQL e i database relazionali sono importanti nel 2021.

Questo perché abbiamo visto molti sviluppatori lottare con i moderni database SQL, ed usarli con riluttanza o fondamentalmente bandirli quasi completamente utilizzando Key-Value o prodotti non SQL.

In assenza di una buona informazione sui punti di forza e debolezza dei server con database SQL, sono nate molte false credenze tra gli sviluppatori. La nostra classe serve per risolvere questo problema.

***Passaggio ai microservizi***

Molte aziende stanno considerando di migrare a micro-servizi. Il nostro punto di vista è che nella stragrande maggioranza dei casi, questa è una esigenza che le aziende vorrebbero avere, ma che in realtà' non hanno.

Data la considerevole potenza di calcolo oggigiorno disponibile dai singoli server, la maggior parte della necessità di calcolo nell'"OnLine Transaction Processing" può essere gestita da una singola macchina. (Per quanto il carico di lavoro analitico è un'altra questione.)

Nella maggior parte dei casi le organizzazioni verrebbero di gran lunga servite meglio mettendo a punto le loro applicazioni in modo da avere il massimo da un singolo server piuttosto che avventurarsi nel

difficile e poco affidabile lavoro di riorganizzare l'architettura del sistema per avere una architettura distribuita.

***Sviluppo in Casa***

Anche quando pensiamo di distribuire in cloud, un approccio che ci piace è quello di sviluppare le applicazioni in casa, su piccoli server locali, nei "vani" aziendali. Senza dubbio questa sembra un'eresia, ma vi sono notevoli vantaggi:

1. Questi server sono essenzialmente gratis da acquistare poiché l'hardware necessario con CPU multiple e grandi memorie è molto economico e ben funzionante al giorno d'oggi.

2. Possiamo gestire i problemi di performance più in fretta, poiché abbiamo un hardware contenuto e possiamo intraprendere in anticipo un'azione correttiva, localmente e senza costi.

3. Minimizziamo il costo dei server del cloud perché conosciamo in anticipo le nostre necessità e possiamo quindi acquistare solo lo stretto necessario.

I backup vengono comunque fatti in cloud per scopi di salvaguardia.

I test possono anche avvenire localmente e successivamente spostarsi nel cloud per il successivo passaggio in produzione.

# Chapter 2 - Amministratore di Database (DBA) di produzione del PostgreSQL

Un Amministratore di Database (DBA) di produzione è responsabile di prendere l'applicazione sviluppata dal team di sviluppatori e fare in modo che l'applicazione operi bene in produzione. Il DBA di produzione verifica le decisioni di progetto prese durante lo sviluppo e le aggiusta come necessario affinché l'applicazione si comporti bene nelle condizioni reali di lavoro.

Ricorda che le applicazioni distribuite nel web possono subire improvvisi aumenti di richieste dovuti a vacanze o inaspettati successi di marketing. Il successo porta a sfide di crescita importanti.

### *DevOps e NetOps*

Dal momento che gli strumenti di gestione sono divenuti più automatizzati, il personale IT può gestire più database ed altri tipi di server rispetto a quanto poteva fare un tempo. Questa tendenza continua fino ai nostri giorni in cui vediamo team di NetOps sempre di più grandi dimensioni.

### *Installazione del Database*

Mentre PostgreSQL e' gratis da scaricare, per installarlo sono necessarie alcune decisioni. Alcuni gestori del pacchetto (p.e. Debian) forniscono un esempio funzionante per farti iniziare, ma questa non è la scelta ottimale per il successivo utilizzo in produzione.

Ci sono anche i diritti dell'utente da analizzare e configurare. PostgreSQL imposta come predefinito un assetto molto conservativo per prediligere la sicurezza: parte della frustrazione che alcuni sviluppatori percepiscono lavorando con PostgreSQL e' dovuta al fatto che non hanno familiarità con i vari settaggi che si possono configurare in modo tale da adeguare la sicurezza preimpostata.

### *Aggiornamento della versione del Database*

Molte aziende rifiutano l'aggiornamento di versione poiché concentrati su altre priorità, oppure diffidano del nuovo software ed in generale hanno un approccio conservativo sull'IT. C'è molto di più da perdere che da guadagnare in un aggiornamento è il loro pensiero.

***Backup Automatici***

Dal momento che i database sono una fonte considerata molto attendibile ("la voce della verità") dall'applicazione, i backup sono essenziali per la maggior parte dei database di produzione. (L'unica eccezione è rappresentata dai database che ricopiamo periodicamente da fonti esterne).

A seconda dell'azienda e dell'applicazione, la perdita dei dati può mettere fine agli affari dell'azienda stessa. Addirittura un fallimento nel ripristinare tempestivamente i dati può avere lo stesso effetto.

Sappiamo e ci aspettiamo che la memoria ed i drive disk, che conservano i dati, possano col tempo guastarsi. E' solo una questione di tempo. Quindi dobbiamo prepararci a questa eventualità predisponendo backup periodici.

Ma anche in questo il DBA (Amministratore di Database) ha un ruolo, deve assicurarsi che la quantità di backup sia minimizzata per rendere efficiente l'uso del database. Uno dei motivi fondamentali per scegliere i database SQL è che prima di tutto essi proteggono dalla perdita di dati e dal danneggiamento dei dati, anche quando l'applicazione o il database dovesse crollare senza preavviso.

(Il rischio di danneggiamento dei dati non è un evento raro oggigiorno, dal momento che gli hard disk sono divenuti così grandi che statisticamente la possibilità di errori di lettura irreversibili interferisce con le operazioni giornaliere.)

Questo in pratica significa che le operazioni del database devono essere ricopiate su un altro server in qualche modo vicino al server del database. Questo spazio e questa larghezza di banda è una risorsa per la quale si deve pagare in un ambiente Cloud; quindi ribadiamo ancora una volta che è necessario un DBA esperto per verificare che gli sviluppatori modifichino una minima quantità di dati per raggiungere i loro obbiettivi; in modo da non travolgere inutilmente con una quantità eccessiva di dati i server di backup.

***Ripristino tramite i Backup di un Database andato in crash***

Mentre i compiti giornalieri di un Amministratore di Database (DBA) di produzione riguardano l'assistenza all'applicazione che usa il database, il compito principale di un DBA è riportare l'applicazione on line dopo un crash.

Grazie all'automazione i moderni ambienti di cloud possono aiutare molto in questa attività, ma il gruppo di DevOps deve fare pratica con queste operazioni; sia per essere certi di avere un valido backup sia per conoscere cosa fare anche sotto stress.

Chi si occupa di DevOps deve anche essere in grado di trasmettere correttamente le informazioni al management circa la tempistica di ripristino e l'eventuale situazione di perdita di qualche dato.

### *Consolidamento del Database*

Una situazione che si presenta comunemente quando non viene coinvolto un DBA è la proliferazione dei database non appena si aggiungono nuove funzionalità alle applicazioni esistenti o avvengono delle espansioni in termini geografici. Dal momento che il database è un "oggetto sconosciuto" e c'è la paura di destabilizzarlo, gli sviluppatori moltiplicheranno i database piuttosto che espanderne con perizia uno già esistente.

Certamente entrambi questi approcci funzionano in termini di servizio fornito all'applicazione, ma un database ben consolidato costa presumibilmente molto meno in termini di risorse cloud.

### *Affinamento delle Performance*

Questo è forse il ruolo maggiormente riconosciuto dell'Amministratore di Database (DBA) di produzione. Il DBA può monitorare le richieste che l'applicazione fa al database e fare di conseguenza degli aggiustamenti. Tutto ciò può avvenire in modo completamente unilaterale senza cambiamenti all'applicazione. In verità in alcuni casi il prodotto viene acquistato ed il codice sorgente non è disponibile (quindi l'applicazione è una scatola chiusa); malgrado ciò è comunque possibile per il DBA eseguire un miglioramento ed una minimizzazione delle risorse.

Il beneficio maggiore si ha quando il DBA è in grado di lavorare con gli sviluppatori e guidarli verso la scrittura di un codice ottimale. Prendendo sulle proprie spalle (al posto degli sviluppatori) il lavoro di progettazione e fornendo loro una struttura dei dati ottimizzata, compatta ed efficiente, il DBA velocizza i tempi di sviluppo ed evita faticosi cicli di tentativi per errore che altrimenti avverrebbero.

### *Qualità dei Dati*

Avere problemi con la qualità dei dati è un'esperienza di quasi tutte le organizzazioni.

Il database è idealmente l'unica fonte di verità nell'applicazione e quindi mantiene lo stato in cui il resto dello stack software è libero di funzionare in modo senza stato. Questa situazione aiuta notevolmente con la scalabilità e la risoluzione dei problemi.

Ma nel caso di database progettati male questo non è sufficiente a correggere gli input e dati non conformi possono venire accettati per poi creare problemi in seguito, quando vengono letti dai vari programmi ed applicazioni.

Il risultato è che chi ha dati inconsistenti deve spendere molto tempo per gestirli all'interno dell'applicazione ripetitivamente invece che aggiustarli una sola volta nel momento in cui vengono inseriti la prima volta. Questa necessità di programmazione correttiva rallenta i programmi del cliente, perché questi devono eseguire molti controlli sulle stringhe di dati per pulire gli input errati invece di gestire dati affidabili ritenendo che ciò che stanno leggendo è già conforme alle specifiche.

Un'altra situazione che produce molti dati problematici è l'unione di due o più database; cosa che accade quando un'azienda acquisisce un concorrente.

La soluzione in questi casi è di evitare la correzione "al volo" fatta dall'applicazione durante l'esecuzione ed applicare invece la correzione una volta per tutte.

Ossia noi preferiamo modificare il database con controlli e vincoli in modo da prevenire il problema e non farlo ripresentare.

# Chapter 3 - Conversione / Migrazione del Database

Dal momento che l'SQL ha avuto molte revisioni dello standard da parte dell'American National Standards Institute (ANSI), le applicazioni possono migrare sovente da un database SQL ad un altro. Il grado di difficoltà' di questa operazione dipende da come hanno agito gli sviluppatori: se si sono attenuti alle caratteristiche di base standardizzate del SQL o se hanno usato le estensioni specifiche del venditore.

In passato i database erano prodotti costosi, disponibili solo per IBM e simili. Ma 50 anni di sviluppo hanno portato a molti database SQL open source molto versatili, di cui PostgreSQL e' certamente il migliore da molti punti di vista.

Di conseguenza molte organizzazioni stanno migrando dai loro database "proprietari", con alti costi di licenza, verso database open source PostgreSQL, un processo una tantum.

Per altri che già stanno usando database open source, come MySQL e simili, il desiderio è di migrare verso un database con migliori caratteristiche, una migliore potenzialità di risorse tecniche ed una comunità di sviluppo più ampia.

***Migrazione da Oracle verso PostgreSQL***

Un grande incentivo a migrare arriva per chi usa il database Oracle, che ha fee ricorrenti annuali molto costose.

La migrazione da Oracle è un processo complicato a volte poichè Oracle anticipa gli standard ANSI, quindi alcuni "isms" utilizzati da Oracle richiedono più volte la riscrittura del codice e un lavoro addizionale per ottenere il risultato corretto.

Tipicamente la migrazione dei dati è un processo che procede senza intoppi, tutto dipende dalla quantità di dati e dal numero di tabelle. In linea di principio ogni cosa può essere letta senza perdita di informazioni e riscritta su PostgreSQL, utilizzando la sua più ampia tipologia di colonne, in modo da archiviare in modo ottimale i dati in ingresso.

Le maggiori difficoltà derivano dal cambiamento del codice dell'applicazione, dal momento che molti presupposti sottostanti i modelli di programmazione differiscono tra il vecchio Oracle ed il più moderno PostgreSQL. Ma in quasi tutti i casi si può arrivare ad un'applicazione che performa bene dopo la migrazione, ed il pagamento delle licenze Oracle può essere interrotto.

# Conclusione

Come può aiutarti Albelissa?

<Existing "Come può aiutarti Albelissa?" text>