Database relazionali e "NoSQL". Alcuni spunti per un corretto inquadramento delle metodologie

Andrea Favretto e Manuela Montagnari ([a]), Marzia Vidulli e Susanna Moser ([b])

([a]) Dipartimento di Studi umanistici, Università di Trieste, afavretto@units.it, montagna@units.it
([b]) Civico Museo d'Antichità "J.J. Winckelmann", Trieste, marzia.vidulli@comune.trieste.it, smoser@units.it

Database relazionali e NoSQL

All'inizio del corrente secolo Garcia-Molina (et alii, 2000) considerava i database essenziali per qualsiasi tipo di attività. L'Autore scriveva che, dalla gestione di prodotti per la loro vendita sul Web (o come supporto a qualsiasi altra attività commerciale), tale metodologia arriva ad essere presente nel nucleo di qualsiasi ricerca scientifica, ad esempio come una rappresentazione dei dati raccolti dagli astronomi, dagli studiosi del genoma umano o dai chimici che esplorano le caratteristiche delle proteine.
Come è noto, un database è una collezione di dati (cfr., fra gli altri: Halpin et al., 2008). Questi, interpretati dall'uomo, possono divenire una raccolta persistente di fatti collegati, ovvero una base informativa. In tale ottica semantica, la metodologia database viene impiegata per produrre una rappresentazione dei dati utile alla loro gestione. Per poterlo fare, è cruciale la fase iniziale di costruzione dello schema concettuale (struttura del database). Lo schema deve garantire tutte le esigenze dell'organizzazione committente. Qualora il contesto nel quale l'organizzazione si muove dovesse cambiare, una struttura database ben fatta deve poter gestire tale cambiamento con correzioni o implementazioni che non pregiudichino il funzionamento di tutte le procedure realizzate prima della variazione del contesto stesso.
Date (1994) sottolinea l'integrazione dei dati quale una delle caratteristiche fondamentali di ogni struttura database ben fatta, al fine di mantenere l'integrità dei dati nel tempo.
Da un punto di vista cronologico, tale metodologia viene fatta risalire a prima dell'era dei computer (Date op. cit.), se si fa riferimento alla gestione delle biblioteche, dei dati anagrafici, medici e delle gestioni contabili di imprese e associazioni varie.
Il modello relazionale per un database viene attribuito a Ted Codd, nel 1970 (Worboys, 2005). Codd, nel suo pionieristico lavoro (1970), aveva parlato di "redundancy" della struttura database, collegata alla necessità di mantenere la base informativa in uno stato di consistenza. Come afferma Berg (et alii, 2013), il modello proposto da Codd cambiò per sempre il modo di pensare i database, grazie al fatto che l'organizzazione logica della base dei dati (lo schema) era scollegata dall'organizzazione fisica dei dati nel computer (una serie di file e indici, la cui gestione è a totale carico del sistema informatico e non all'ideatore dello schema logico). Questo ne decretò il grande successo e

ne fece il principio standard per tutti i sistemi database. Fra il 1974 ed il 1977 furono creati due importanti database, "INGRES" e "R", che hanno permesso lo sviluppo nel tempo di due software di gestione (DBMS – Database Management System) oggi molto diffusi: Postgres (open source - https://www.postgresql.org) e Oracle (commerciale - https://www.oracle.com/index.html). INGRES fu sviluppato presso l'università della California-Berkeley mentre R presso l'IBM. In questo periodo si consolidò anche SQL (Structured Query Language) il linguaggio per interfacciarsi alle strutture database.
A metà degli anni '80 nacque il concetto di Object Oriented Database System (OODBMS-cfr.: Prbhu, 2011). Tale tipo di struttura è in grado di registrare e gestire l'informazione sotto forma di oggetti, al pari dell'omonimo paradigma di programmazione che, nello stesso periodo, vide nascere uno dei suoi maggiori esponenti, ovvero il linguaggio C#. L'approccio OO deriva dal modello cosiddetto "Entity-Relationship Attribute" (ERA). Nella metà degli anni '70 lo standard ERA aveva arricchito i database relazionali con la modellizzazione della realtà da meccanizzare in una serie di entità, unità logiche informative che nella struttura database divengono le tavole da mettere in relazione (Chen, 1976). OODBMS viene usato per gestire dati complessi e diversi fra loro, quali ad esempio i dati grafici legati al territorio (in collegamento con i GIS – Geographical Information System).
Gli anni '90 videro il consolidamento del Web, che impose l'architettura client-server di Internet a tutti i livelli delle applicazioni informatiche. I DBMS orientati agli oggetti divennero relazionali (ORDBMS - Object Relational Database Mnagement System – cfr.: Brown, 2000), per poter utilizzare appieno tale modello senza rinunciare alla capacità degli oggetti di incapsulare diversi tipi di dati (distribuiti via Web).
Lo sviluppo esponenziale del Web negli ultimi 20 anni, attraverso la crescita rapidissima di società "Web oriented" quali Amazon, Google, Twitter e Facebook, ha evidenziato la necessità di gestire immense quantità di dati non omogenei, memorizzati in diversi server sparsi in Internet. Nel 2006, 2007 e 2010 rispettivamente Google, Amazon e Facebook hanno pubblicato degli articoli ("Bigtable", "Dynamo" e "Cassandra") che descrivono le soluzioni adottate per la gestione informatica dei dati. Le pubblicazioni descrivono strutture database alternative al modello relazionale, che sacrificano in parte la consistenza del dato in favore di una maggiore disponibilità dello stesso[1].
Il termine NoSQL precede le citate pubblicazioni di Google, Amazon e Facebook e si deve a Carlo Strozzi, uno sviluppatore informatico che nel 1998 creò un database relazionale open source senza un'interfaccia SQL (Berg et al., op. cit.). La metodologia NoSQL non viene però attribuita ad un singolo sviluppatore ma a gruppi indipendenti, che via via cercarono di risolvere i problemi di gestione dei loro dati e non trovarono nel modello relazionale una valida soluzione. Oggi ci sono diverse applicazioni database NoSQL.

[1] La consistenza del dato fa parte delle quattro proprietà che garantiscono l'affidabilità delle transazioni del database. ACID sta per l'appunto per Atomicity (le transazioni devono venir fatte completamente e non in parte), Consistency (le transazioni possono variare i dati solo sotto determinate regole), Isolation (determina la visibilità dei risultati di una transazione parziale – ad esempio l'inserimento solo di alcuni campi in un record e la loro visibilità a tutti gli utenti che si connettono al database), Durability (la sicurezza delle transazioni che devono produrre risultati salvati permanentemente).

Fowler (2015) identifica quattro "core features" per NoSQL:

1. "Schema agnostic". Non è richiesto uno schema dei dati.
2. "Nonrelational". Al posto delle relazioni i singoli record contengono tutte le informazioni, che possono anche differire in qualità e quantità (ad esempio: due indirizzi di spedizione invece di uno per un determinato cliente/record).
3. "Commodity hardware". La scalabilità dell'hardware non è più verticale ma orizzontale (non più server ultra configurati ma molte macchine più economiche collegate fra loro).
4. "Highly distributable". Si possono quindi usare gruppi di server per gestire un database con molti dati.

Di conseguenza una possibile definizione di DBMS NoSQL è:

1. Ogni record creato non deve seguire uno schema predefinito (ovvero la struttura a campi/colonne della tabella).
2. La base dei dati viene memorizzata su hardware distribuito e a media configurazione (il cosiddetto "commodity hardware").
3. La base dei dati non usa il modello relazionale.

Ci sono quattro diversi tipi di dati (cosiddetti "data types") per i database NoSQL, che corrispondono ad altrettanti metodi di memorizzazione e gestione dei dati digitali (cfr.: AmitPal, 2016; Nayak et al., 2013):

1. "Key value store". Ad un valore chiave va fatta corrispondere una determinata tipologia di dati (String, Json, BLOB).
2. "Document". Ad un valore chiave viene fatto corrispondere un documento (anche in formato testuale-non strutturato).
3. "Column-oriented". Ad un valore chiave viene associato uno o più attributi che non stanno su una tavola ma su un'architettura distribuita su più computer.
4. "Graph". I dati sono memorizzati utilizzando la teoria dei grafi. Si tratta di nodi e collegamenti ("edge") ove i nodi sono gli oggetti (entità) e i collegamenti le relazioni fra questi.

Si desidera presentare un piccolo contributo conoscitivo presentando un esempio di struttura database declinata secondo il modello relazionale e NoSQl (in modalità: "document data type").
Viene presentato un semplice database costruito a fini didattici durante le lezioni di Paletnologia e GIS per l'Archeologia, del corso di laurea triennale di Lettere antiche e moderne, arti, comunicazione del Dipartimento di Studi umanistici/Università di Trieste – a.a. 2019-2020. Il database registra informazioni collegate ad alcuni reperti archeologici conservati nel Civico Museo d'Antichità "J.J. Winckelmann" di Trieste, frutto degli scavi di Carlo Marchesetti del 1889 nella Grotta Tominz (zona di San Canziano del Timavo – attuale Slovenia). La medesima raccolta di reperti viene organizzata utilizzando gli applicativi Ms Access e MongoDB, rispettivamente per il metodo relazionale e quello NoSQL (document oriented).
Dopo un breve inquadramento del complesso archeologico usato come caso di studio, vengono presentate brevemente le due diverse strutture. Alcune considerazioni conclusive chiudono il lavoro.

I reperti archeologici della grotta Tominz

… la più grande e spaziosa delle caverne laterali [del complesso di S. Canziano]*, che in lunghezza misura non meno di 290*^{m}*… Di facilissimo accesso, essa ci presenta un magnifico vestibolo volto a settentrione, largo da 25 a 30 m. ed alto da 10 a 15, che si estende per quasi 150 m. … Verso l'estremità del vestibolo la caverna possede una spaziosa galleria laterale che mette a varie stanze…* Così Carlo Marchesetti descrive la grotta Tominz o Preistorica, che fa parte del complesso di gallerie e cavità lungo oltre 5 km scavato nel corso di milioni d'anni dal fiume Timavo, che qui, nel Carso sloveno a poca distanza da Trieste, s'inabissa per riemergere in superficie 34 km a valle. Marchesetti – una delle figure-chiave della nascita della ricerca pre-protostorica nelle regioni dell'Alto Adriatico fra fine '800 e inizi '900, oltre che botanico di fama europea – scavò in questa e in altre grotte della zona: i materiali raccolti furono conservati presso il Museo Civico di Storia Naturale di Trieste, di cui egli fu direttore dal 1876 al 1921, e dopo la sua morte, avvenuta nel 1926, furono trasferiti (tranne i resti umani e faunistici – cfr. Betic, Bernardini, 2003) al Civico Museo d'Antichità della stessa città. Lo studioso pubblicò una relazione abbastanza dettagliata sui risultati delle indagini (Marchesetti, 1889), ma lasciò anche appunti e disegni nei suoi taccuini (tuttora largamente inediti, conservati nella Biblioteca Civica di Trieste – cfr. Montagnari Kokelj 1994), incluso uno schizzo della sequenza stratigrafica identificata nella grotta Tominz.
La possibilità di confrontare la documentazione edita e inedita e i reperti è fondamentale ai fini del progetto di riesame sistematico e studio di questo complesso, proposto dal Museo Regionale di Capodistria e dal Park Škocjanske Jame ai Civici Musei di Trieste e avviato nel 2018. All'interno del progetto è stato affidato all'Università di Trieste lo studio dei materiali litici: nella fase preliminare d'inventariazione e catalogazione sono stati coinvolti gli studenti del corso di Paletnologia (laurea triennale); gli stessi, frequentando anche il corso di GIS per l'Archeologia, hanno poi inserito i dati nel database costruito *ad hoc*.

I materiali presi in esame provengono dallo *strato più profondo, grosso 20 a 30 cent. che giaceva da 1 a 3 metri sotto l'odierno livello,* [e che] *conteneva numerosi istrumenti di pietra e d'osso con grande numero di cocci e di resti d'animali. Vi preponderano oggetti di selce, che a quanto pare venivano fabbricati nella stessa caverna, dappoiché copiosi vi sono i nuclei e le relative scheggie di rifiuto* (Marchesetti 1889 – cfr.: fig. 1).

Figura 1 - Strumenti in selce rinvenuti nella grotta Tominz (Marchesetti 1889, tav. I [parziale]).

Nuclei, scarti di fabbricazione e manufatti non ritoccati sono stati identificati ma non inseriti nel database, limitato in fase preliminare agli strumenti: ne sono stati

riconosciuti una ventina, analizzati in base alle tipologie più comunemente usate per le industrie della Preistoria recente (Laplace, 1968; Bagolini, 1970). Nonostante ovvie differenze fra la descrizione dei reperti fatta da Marchesetti e quella attuale, dovute agli sviluppi dell'archeologia in generale e degli studi tecno-tipologici della litica in particolare, l'inquadramento dei pezzi e dello strato più profondo del deposito fra Neolitico ed *epoca eneo-litica* data dallo studioso è confermata già nella fase iniziale dello studio in corso.

Le due strutture database

Come si è detto, si è realizzata una semplice struttura database, declinata nel modello relazionale (utilizzando l'applicativo Ms. Access), e in quello NoSQL (utilizzando l'applicativo libero MongoDB).

La fig. 2 mostra la struttura del database relazionale realizzata con Ms. Access.

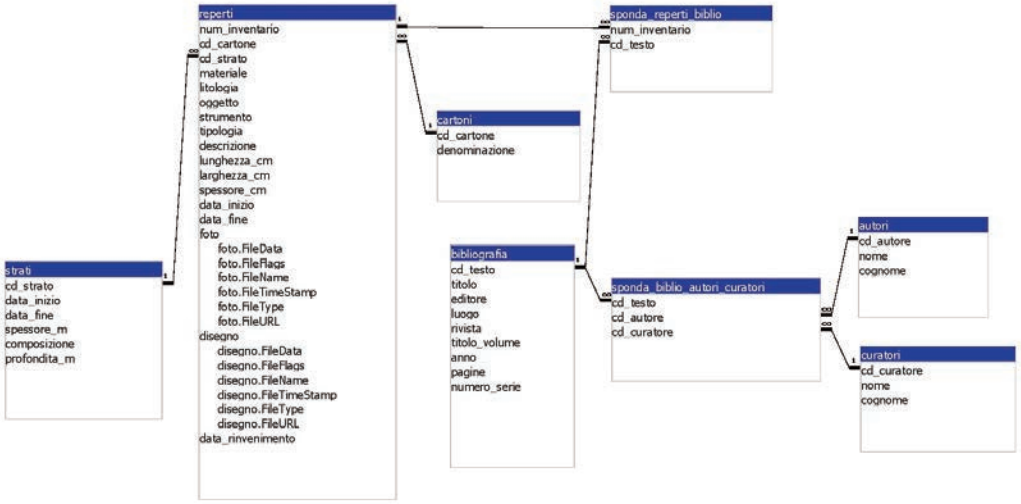

Figura 2 – La struttura del database relazionale sviluppato (Ms. Access).

Come si può vedere, si tratta di 8 tavole unite da relazioni "uno sta a molti" e "molti stanno a molti":

1. La tavola "reperti" raccoglie le informazioni relative al singolo ritrovamento e si collega alle tavole "strati" e "cartoni" con relazioni uno a molti mentre il collegamento alla tavola "bibliografia" (relazione molti a molti) è realizzato dalla tavola di sponda denominata "sponda_reperti_biblio".
2. La tavola "strati" memorizza le informazioni relative agli strati scavati nella grotta, nei quali i singoli reperti sono stati ritrovati (uno strato, molti reperti).
3. La tavola "cartoni" è dedicata ai vari supporti di cartone sui quali il Marchesetti aveva fissato i reperti da lui rinvenuti (un cartone molti reperti).
4. La tavola "bibliografia" riporta le informazioni sui testi cui possono fare riferimento i singoli reperti (un reperto molti testi ma anche un testo molti reperti).
5. La tavola "sponda_reperti_biblio" serve per costruire la relazione molti a molti fra le tavole "reperti" e "bibliografia".
6. La tavola "autori" memorizza le informazioni sugli autori che hanno scritto i testi riportati sulla tavola "bibliografia", con la quale è collegata mediante una relazione molti a molti.

7. La tavola "curatori" memorizza le informazioni sui curatori che hanno curato i testi riportati sulla tavola "bibliografia", con la quale è collegata mediante una relazione molti a molti.
8. La tavola "sponda_biblio_autori_curatori" serve per realizzare le relazioni molti a molti con le tavole "bibliografia" e "autori" (un testo molti autori ma anche un autore molti testi) e con le tavole "bibliografia" e "curatori" (un testo molti curatori ma anche un curatore molti testi).

La medesima struttura è stata anche sviluppata utilizzando MongoDB. Il cosiddetto "document data model" adottato da MongoDB presenta il documento come una singola struttura (corrisponde al record nell'impostazione relazionale), che può contenere dati connessi sotto forma di sotto-documenti inseriti, in tal modo evitando la presenza delle tavole collegate da relazioni (uno a molti, molti a molti, ecc.), tipiche del modello SQL. I documenti sono compresi nelle collezioni (che corrispondono alle tavole). Un database può quindi essere formato da una o più collezioni, a loro volta formate da tanti documenti (cfr.: MongoDB, 2019). Se nella struttura DB ci sono più collezioni, queste sono collegate mediante relazioni uno a molti (come nell'impostazione SQL). Naturalmente l'altro modo di collegare dei dati è quello di tenerli inseriti (embedded) nei singoli documenti. Quando utilizzare un metodo o l'altro dipende dalle caratteristiche dei dati.

Figura 3 – Esempio di query per vedere ciascun reperto con il suo strato di riferimento

Si è optato per due collezioni di documenti, "reperti" e "strati", collegati da una chiave comune (primaria in "strati" e secondaria in "reperti"), con la quale si è costruita una relazione uno a molti fra le due collezioni. Il collegamento fra i reperti e la bibliografia di appoggio è stato invece realizzato inserendo in ciascun documento della collezione "reperti" tutte le informazioni sui testi collegati. Il motivo di tale scelta è stato che lo strato cui si riferisce il reperto è sempre uno mentre i testi possono essere più d'uno. Non solo, ci possono essere reperti che hanno più testi collegati ed altri che ne hanno solo uno. In tal modo si può sfruttare la possibilità dei database NoSQL di avere documenti diversi l'uno dall'altro.
La fig. 3 mostra il risultato di una query per vedere ciascun reperto con il suo strato di riferimento (visualizzazione in modalità JSON Viewer - applicativo Studio 3T – GUI per MongoDB - https://studio3t.com).

Considerazioni conclusive

In questa breve nota si è cercato di offrire alcuni spunti di riflessione sul modello database NoSQL orientato ai documenti attraverso il confronto con quello relazionale sulla base di un caso concreto. L'occasione è stata fornita

dalle attività didattiche dei due corsi universitari di Paletnologia e GIS per l'Archeologia, del Dipartimento di Studi umanistici dell'Università di Trieste. E' stato costruito un database relazionale in aula, utilizzando l'applicativo Ms. Access. Tale database è stato poi "tradotto" in modalità NoSQL, utilizzando l'applicativo MongoDB. L'esperienza è stata molto positiva da un punto di vista didattico.
Di seguito si desidera proporre alcune riflessioni desunte dal lavoro svolto.
Il modello relazionale ha dalla sua un notevole livello di perfezionamento degli applicativi DBMS in circolazione, data la sua "maggiore età" rispetto al concorrente. L'integrità dei dati permette poi una notevole razionalizzazione della struttura e garantisce una maggior consistenza dei dati all'interno del database.
NoSQL d'altra parte permette di semplificare la costruzione della struttura, rendendo più immediata la comprensione della base dei dati e, di conseguenza, non necessario il preventivo disegno di complessi e macchinosi schemi. Tutto ciò viene pagato con l'allungamento dei tempi di inserimento e aggiornamento dei dati (a causa della duplicazione degli stessi nei vari documenti). Inoltre c'è un maggior rischio di incongruenza e quindi inconsistenza dei dati, a causa della mancanza del vincolo di integrità referenziale.
Il modello relazionale si appoggia al linguaggio SQL per la gestione della base dei dati. Tale linguaggio è molto efficiente, compatto e di facile comprensione anche ai non esperti di programmazione. MongoDB comunica con l'esterno attraverso specifici linguaggi di programmazione, che devono essere scaricati separatamente (si veda a tal riguardo: https://docs.mongodb.com/ecosystem/drivers/) e che possono risultare ostici per degli utilizzatori non esperti di informatica. Esistono delle interfacce grafiche, alcune libere (MongoDB Compass) ed alcune a pagamento (Studio 3T) ma queste non garantiscono la fattibilità di tutte le operazioni possibili sulla base dei dati che i comandi a console invece permettono (accesso tramite riga di comando – Mongo shell - https://docs.mongodb.com/manual/mongo/).
In conclusione, il lavoro svolto in modalità NoSQL ha evidenziato un prodotto dalle grandi potenzialità, adatto a gestire dati eterogenei che non necessitino di una struttura fissa, tipica invece del modello relazionale. La flessibilità della base dei dati, che può adattarsi dinamicamente a variazioni del contesto esterno e la sua scalabilità sono forse pregi non particolarmente necessari ad un progetto quale quello presentato.
Ms. Access, con la sua facilità di utilizzo (ad esempio nella gestione di contenuti diversi dal testo, cfr. i campi allegati) è particolarmente adatto ad un uso didattico, in quanto permette di arrivare velocemente ad un prodotto che funziona, invogliando così gli studenti ad approfondire l'argomento.

Riferimenti bibliografici

Amazon – DeCandia G., Hastorun D., Jampani M., Kakulapati G., Lakshman A., Pilchin A., Sivasubramanian S., Vosshall P., Vogels W. (2007). "Dynamo: Amazon's Highly Available Key-value Store", *SOSP '07 ACM 978*, Washington, https://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf.

AmitPal P. (2016). "A Review of NoSQL Databases, Types and Comparison with Relational Database", *International Journal of Engineering Science and Computing*, 6/5.
Bagolini B. (1970). "Ricerche tipologiche sul gruppo dei foliati nelle industrie di età olocenica della Valle Padana", *Annali dell'Università di Ferrara*, sez. 15, 1/11.
Berg K. L., Seymour T., Goel R. (2013). "History of Databases", *International Journal of Management & Information Systems*, 17/1.
Betic A., Bernardini F. (2003). "Reperti archeologici provenienti da San Canziano del Timavo (Slovenia) ritrovati nel Museo Civico di Storia Naturale di Trieste (studio preliminare)", *Atti del Museo Civico di Storia Naturale*, 50.
Brown P. G. (2000). *Object-Relational Database Development: A Plumber's Guide with Cdrom*, Prentice Hall Upper Saddle River, NJ.
Chen P. P-S. (1976). "The Entity-Relationship Model – Toward a Unified View of Data", *ACM Transactions on Database Systems*, 1/1.
Codd T. (1970). "A Relational model of Data for Large Shared Data Banks", *Communication of the ACM*, 13/6.
Date C. J. (1994). *An Introduction to Database Systems*, Addison Wesley, New York.
Facebook – Lakshman A., Malik P. (2010). "Cassandra: a decentralized structured storage system", *ACM SIGOPS Operating Systems Review*, 44/2.
Fowler A. (2015). *NoSQL for Dummies*, Wiley & Sons, New Jersey.
Garcia-Molina H., Ullman J. D., Widom J. (2000), *Database System Implementation*, Prentice Hall, New Jersey.
Google Inc. – Chang F., Dean J., Ghemawat S., Hsieh W. C., Wallach D. A., Burrows M., Chandra T., Fikes A., Gruber R. E. (2006). "Bigtable: A Distributed Storage System for Structured Data", *OSDI*, https://static.googleusercontent.com/media/research.google.com/it//archive/bigtable-osdi06.pdf .
Halpin T., Morgan T. (2008). *Information Modeling and Relational Databases*, Elsevier, New York.
Laplace G. (1968). "Recherches de typologie analytique", *Origini*, 2.
Marchesetti C. (1889). "Ricerche preistoriche nelle caverne di S. Canziano presso Trieste", *Bollettino della Società Adriatica di Scienze Naturali in Trieste*, 11.
MongoDB. (2019). *MongoDB Architecture Guide: Overview*, https://www.mongodb.com/collateral/mongodb-architecture-guide .
Montagnari Kokelj E. (1994). "Carlo Marchesetti fra Preistoria e Protostoria". In: E. Montagnari Kokelj (a cura di), *Atti della giornata internazionale di studio su Carlo Marchesetti*, Trieste, 9 ottobre 1993, Civici Musei di Storia ed Arte, Trieste.
Nayak A., Poriya A., Poojary D. (2013). "Type of NOSQL Databases and its Comparison with Relational Databases", *International Journal of Applied Information Systems*, 5/4.
Prabhu C. S. R. (2011). *Object-Oriented Database Systems: Approaches and Architectures*, PHI learning, New Delhi.