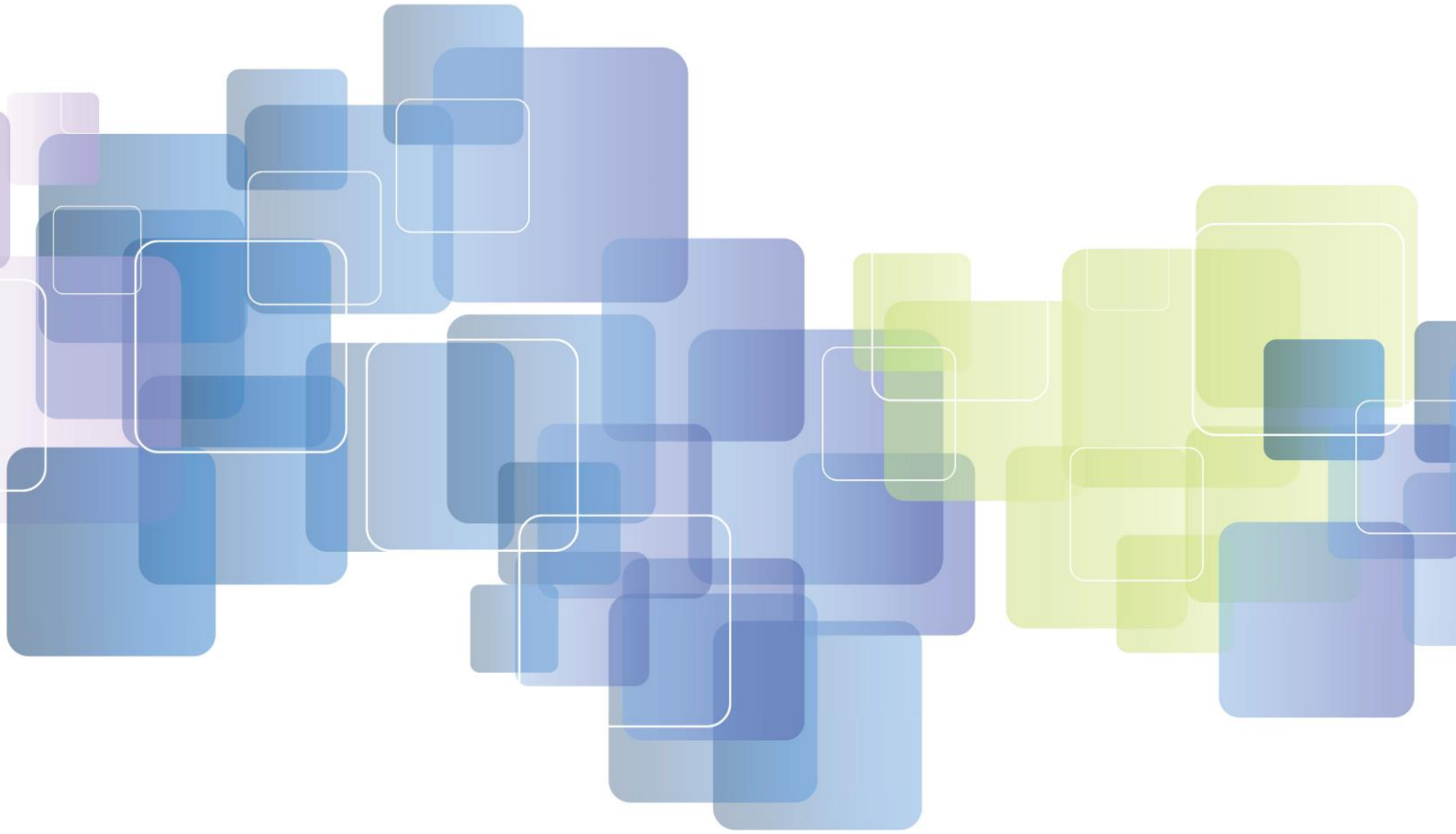
TecnetDati

# Alla ricerca Dell'oggetto perduto

**Emilio C. Porcelli**



# Cento, mille volte più veloci! Così gli ODBMS rispetto ai DBMS relazionali. Ma è tutto oro ciò che luccica?

Dopo tutte le battaglie che lo hanno visto prevalere sui DBMS gerarchici e a rete, il Relazionale comincia a confrontarsi sul mercato con una nuova generazione di database, quelli orientati agli Oggetti. A chi andrà questa volta la vittoria? I DBMS OO vantano buone frecce al loro arco. Promettono, tra l'altro, prestazioni di interi ordini di magnitudine superiori a quelle dei DBMS relazionali. Ma è tutto oro ciò che luccica? Il vostro Autore ne ha discusso nel corso di una tavola rotonda al DB Forum 96, il VI Convegno Internazionale sullo stato dell'arte del mondo dei DBMS e delle loro applicazioni che si è svolto a Milano dal 12 al 14 novembre. Ne discutono su queste pagine, in maggiore dettaglio e con accenti un po' diversi, anche due nostre vecchie conoscenze, Alfa e Omega

Alfa - Cosa stai facendo in pieno giorno con una lanterna accesa in mano?

Omega- Faccio come Diogene. Lui cercava l'Uomo, io cerco l'Oggetto.

Alfa - Ma sei sicuro che una lanterna sia l'attrezzo più indicato per trovarlo? Fossi in te mi procurerei piuttosto un bell'Identificatore.

Omega- Figurati se non c'ho pensato. Ma è proprio qui che nasce il problema. Come dev'essere questo Identificatore, un valore come per esempio un codice cliente o un numero di conto corrente? Oppure un non-valore come vorrebbero invece i fautori dell'Object- Oriented?

Alfa - Un non-valore? Spiegati meglio.

Omega - Facciamo allora un piccolo passo indietro. Oggi nel campo dei DBMS si confrontano due scuole di pensiero: da una parte ci sono i "rivoluzionari", dall'altra gli "evoluzionisti". Del resto ne abbiamo già parlato (cfr. "Troppi Manifesti!" su CWI del 27 febbraio 1996). Ricordi?

Alfa - Ricordo perfettamente. I "rivoluzionari" del database, raccolti sotto la bandiera dell'Object- Oriented, si sono riuniti a Kyoto in Giappone nel novembre del 1989 e hanno dato alla luce il loro "Manifesto dei Sistemi Database Orientati agli Oggetti". Per non essere da meno gli "evoluzionisti", e cioè i sostenitori della estensione in senso Object-Oriented dei DBMS relazionali, qualche mese dopo si sono a loro volta riuniti e hanno pubblicato il "Manifesto dei Sistemi Database della Terza Generazione". Non mi dire che anche in fatto di Identificatori le loro posizioni divergono.

Omega- Dire che divergono è dir poco. La materia del contendere infatti è tutta qui.

## Lo diceva anche Kant

Alfa - Qual è il punto di vista degli "evoluzionisti"?

Omega - E' molto semplice. Come sai un DBMS relazionale viene percepito dai suoi utenti come una collezione di tabelle e nulla più. Inoltre ogni tabella è popolata esclusivamente da valori tratti dai suoi domini. Infine l'accesso e la manipolazione di questi valori può avvenire solo attraverso la famiglia degli operatori relazionali. Quindi in un DBMS relazionale, o RDBMS, gli Identificatori, o Primary Key (PK), sono sicuramente dei valori.

Alfa - Invece i "rivoluzionari" la pensano diversamente.

Omega - Proprio così. Per loro usare degli Identificatori basati sui valori significa fare una pericolosa confusione. Già nel corso di OOPSLA 86, la Conferenza sui Sistemi e i Linguaggi Orientati agli Oggetti che quell'anno si svolse a Seattle, c'era chi affermava: "Numerosi sistemi database usano chiavi identificative per distinguere oggetti persistenti, confondendo così valore dei dati e identità".

Alfa - E per non cadere in questa confusione cosa propongono i fautori dei database Object-Oriented?

Omega - C'è un punto di vista per così dire "ufficiale" che si riassume in un'affermazione del loro Manifesto: "In un sistema basato sull'identità, un oggetto è indipendente dai suoi valori". Quindi in un DBMS Object-Oriented, o ODBMS, aspettiamoci di trovare degli Identificatori degli Oggetti, o Oid, generati automaticamente dal sistema che saranno tutto fuor che dei valori.

Alfa - Adesso capisco il perchè delle tue perplessità: se gli Oid non sono dei valori, sono dei non-valori. E a questo punto mi sorge il sospetto che adottare come Identificatore un non-valore non sia senza conseguenze.

Omega- Le conseguenze ci sono e sono tante. La prima e più importante, per dirla con Martin e Odell, è che un oggetto esiste indipendentemente dalle sue proprietà.

Alfa - Anche della sua proprietà di esistere?

Omega - Anche di quella. Anzi, l'esistenza non è una proprietà.

Alfa - E chi l'ha detto?

Omega - Nientemeno che Immanuele Kant nella sua "Critica della Ragion Pura". Ecco come argomenta il grande filosofo: "L'esistenza non è una proprietà. Quando ascriviamo un attributo a qualcosa, implicitamente affermiamo che questo qualcosa esiste; cosicchè se l'esistenza stessa fosse un attributo, tutte le proposizioni affermative di esistenza sarebbero delle tautologie, e tutte quelle negative sarebbero contraddittorie in sè il che non è". Dire uguale non è come dire identico

Alfa - Dopo Diogene, Kant. Confesso che, discorrendo di queste cose, non mi sarebbe mai passato per la testa di dovermi imbattere in personaggi del genere.

Omega - Temo che gli incontri non finiranno qui. Ma veniamo a una seconda conseguenza della nozione di Oid e cioè alla curiosa distinzione tra oggetti uguali e oggetti identici. Ne parlano due professori, Elisa Bertino e Lorenzo Martino, in un loro recente libro (Sistemi di Basi di Dati Orientati agli Oggetti, Addison-Wesley Masson 1992) che è stato poi pubblicato anche negli Stati Uniti. Non è cosa che capiti tutti i giorni.

Alfa - E quale sarebbe questa distinzione?

Omega - Due oggetti sono identici se sono lo stesso oggetto, se hanno cioè lo stesso Oid; sono invece uguali se i valori di tutti i loro attributi sono ricorsivamente uguali. Va da sè - aggiungono i nostri due professori - che due oggetti identici sono anche uguali, mentre non è vero il contrario.

Alfa - Non ho nessun problema a figurarmi due oggetti del tutto uguali come due mele, due monetine da cento lire o anche due gemelli omozigoti. Trovo invece serie difficoltà a immaginare due oggetti identici. Del resto dire che un oggetto è uguale a se stesso non è una banalità? E dire che è identico a qualcosa d'altro non è una assurdità? Mi chiedo allora che utilità abbia parlare di identità.

Omega - In questo sei in buona compagnia, infatti la stessa domanda se la poneva anche Wittgenstein. Ma una utilità c'è. Chiediti piuttosto: ci si può bagnare due volte nelle acque dello stesso fiume?

Alfa - Sembrerebbe una massima Zen.

Omega - Ma non lo è. A sollevare il problema è stato Eraclito qualcosa come duemila e cinquecento anni fa.

Alfa - Ci risiamo con i filosofi...

Omega - Eccome se ci risiamo! Ma rispondi piuttosto alla mia domanda.

Alfa - Fammici pensare. So che ci si può bagnare due volte. Anzi no, non ci si può bagnare due volte perché, - così argomentava Eraclito - il fiume rinnova in continuazione la sua sostanza e quindi non sarà mai lo stesso. Vedi, anch'io ho qualche buona lettura.

Omega - Non ne dubitavo affatto, anche se non me la sentirei di sposare fino in fondo la tesi di Eraclito. Infatti affermando l'identità costante nel tempo di un fiume, o di qualsiasi altro oggetto, in genere non pensiamo alla ritenzione della sua sostanza ma ad altre continuità. Dire per esempio che il Signor Rossi è nato cinquant'anni fa a Roma e che oggi abita vicino alla Stazione Centrale di Milano significa identificare quel Rossi con l'infante venuto alla luce a Roma cinquant'anni fa (e in tutto questo tempo la sua sostanza corporea si è sicuramente rinnovata) e la sua casa con una di quelle vicine alla Stazione. Identità come queste permeano i nostri discorsi quotidiani. Anche quelli che vengono fatti nel Database.

## Invarianza e unicità

Alfa - Mi hai convinto. Parlare di identità è sicuramente utile, anche a proposito di Database.

Omega - Ma allora, se vogliamo davvero preservare la continuità nel tempo del Signor Rossi e metterlo in relazione con la casa in cui abita, il nostro Identificatore dovrà possedere due caratteristiche ben precise.

Alfa - E sarebbero?

Omega - L'invarianza nel tempo e l'unicità Almeno su questo i fautori degli RDBMS e degli ODBMS sono d'accordo. Ma i "rivoluzionari" non si fermano qui e infatti aggiungono: l'Identificatore dev'essere anche univoco nel sistema. Ora ti chiedo: chi è in grado di assicurare, oltre che l'invarianza, anche l'unicità degli Identificatori in tutto il sistema?

Alfa - La risposta è facile: il sistema stesso.

Omega - Tutto il contrario di ciò che accade negli attuali RDBMS dove le PK vengono assegnate dagli utenti in modo dichiarativo. E se questi utenti sono, come spesso accade, decine o centinaia, e le tabelle nel database migliaia o decine di migliaia, quali garanzie avremo in concreto che le PK siano univoche nel sistema?

Alfa - A occhio poche, o meglio nessuna. Tutto dipende dalla fortuna o dalla buona volontà delle persone. Ma far conto sulla fortuna o sulla buona volontà, che sarà anche una dote evangelica ma ha poco a che vedere con la tecnologia del database, per garantire l'univocità delle PK sembrerebbe per lo meno azzardato.

Omega - E' esattamente uno degli argomenti che i fautori degli ODBMS, ossessionati come sono dallo spettro della polistanzazione, sventolano per vantare la superiorità dei loro Oid sulle PK. Il Modello nel cassetto

Alfa - Ha l'aria di essere un solido argomento.

Omega - Solido sì, ma solo fino a un certo punto. Infatti il Modello Relazionale, o meglio una sua particolare versione, non solo prevede l'unicità delle PK nel sistema ma fornisce anche i mezzi per assicurarla.

Alfa - E quale sarebbe questa particolare versione del Modello?

Omega - E' l'RM/T che Edgar Codd, l' "inventore" del Modello Relazionale, ha proposto in tempi non sospetti e cioè nel 1979.

Alfa - Cosa proponeva Codd nel lontano 1979?

Omega - Che le PK assegnate in modo dichiarativo dagli utenti venissero "surrogate", e cioè sostituite o anche solo affiancate, da identificatori generati automaticamente dal sistema e in quanto tali univoci nel database e invarianti nel tempo. Gli utenti possono chiedere al database di generare o cancellare questi "surrogati", ma non hanno alcun controllo sui loro valori che del resto non possono nemmeno vedere.

Alfa - Ma dire che un valore non è visibile non è come dire che è un non-valore? I tuoi surrogati sembrerebbero la copia conforme degli Oid.

Omega - Non saranno visibili, ma restano pur sempre dei valori. Sono infatti basati su uno speciale dominio che Codd chiama dominio-E ed è la fonte di tutti i surrogati del sistema. Gli attributi che traggono i loro valori da questo dominio, detti a loro volta attributi-E, possono venire manipolati solo attraverso i consueti operatori relazionali e a essi si applicano anche i meccanismi dell'integrità referenziale. Va da sé, che l'introduzione del dominio-E, degli attributi-E e dei surrogati non decreta necessariamente la fine delle PK assegnate in modo dichiarativo. Se lo desidera l'utente potrà tranquillamente continuare a usarle, ma potrà anche farne a meno: per specificare i suoi (equi) Join gli basterà citare l'intestazione di quella colonna in più che si ritroverà nelle sue tabelle. E' la colonna che corrisponde agli attributi-E i cui valori non sono visibili.

Alfa - Tutto questo però solo sulla carta. Non mi risulta infatti che ci sia oggi al mondo un RDBMS che si conformi con l'RM/T.

Omega - E' vero, ma non tutte le idee di Codd sono state lasciate cadere. Infatti lo standard SQL3, ancora in via di elaborazione, prevede che l'identità degli oggetti (ovvero righe di tabelle relazionali) possa essere anche assicurata da identificatori univoci e invarianti generati dal sistema che si rendono visibili nella prima colonna di ciascuna tabella. L'unica differenza rispetto ai surrogati dell'RM/T sembrerebbe la loro visibilità

Alfa - Il buon Dott. Codd è uno che ha saputo vedere lontano, peccato che tante sue proposte siano rimaste a lungo inascoltate.

Omega - E infatti il suo RM/T è finito in un cassetto. In attesa di tempi migliori, per dirla sempre con Codd.

## La sindrome del Grande Fratello

Alfa - E nell'attesa che i tempi siano più propizi, ecco farsi avanti gli ODBMS con i loro Oid che non sono dei valori e vengono generati automaticamente dal sistema. Chissà quanto sono lunghi.

Omega - Sulla lunghezza degli Oid le due maggiori autorità in materia, il Comitato X3H2 dell'ANSI e l'Object Management Group (OMG), non hanno voluto prendere posizione. Potranno essere quindi stringhe di 32, 64 o 128 bit, oppure di lunghezza variabile. Ma a quanto scrive su una rivista consorella un altro professore, Michele Marchesi, oggi ci sarebbe un generale consenso su una lunghezza di 96 bit.

Alfa - Perchè mai 96 bit?

Omega - Confesso che non lo so. Ma lo dice un Professore e ai Professori, così come ai Filosofi, è bene dar retta.

Alfa - Lasciami allora fare quattro conti. Con 96 bit a disposizione lo spazio potenziale di indirizzamento è rappresentato da un numero molto vicino a 8 seguito dalla bellezza di

ben 28 zeri. Con tanta abbondanza l'identità degli oggetti nel database sembrerebbe garantita.

Omega - E invece potrebbe non bastare. C'è infatti chi pretende di identificare con un Oid di 96 bit non solo tutti gli oggetti nel database, ma anche tutti quelli che popolano un ambiente multidatabase, tutti i DBMS che vi partecipano e per finire anche i nomi dei loro produttori.

Alfa - Ma è la sindrome del Grande Fratello! Non mi sentirei a mio agio se mi trovassi a lavorare su un database locale che avesse la pretesa di sapere sempre tutto di tutti i database remoti con i quali è collegato. Senza contare che così l'ODBMS finirebbe con l'usurpare le funzioni che si richiedono normalmente a un buon middleware.

Omega - Hai perfettamente ragione. Anch'io ho almeno tre buone ragioni per dubitare della bontà di idee come queste. La prima è che può nascere una pericolosa confusione tra indirizzabilità ed enumerabilità Qui i nostri amici filosofi mi darebbero ragione: se è vero infatti che il database viene costruito a immagine del Mondo Reale, non solo si finirebbe con l'assumere che gli oggetti del Mondo Reale sono enumerabili, il che può essere anche ragionevole, ma anche che il loro numero è finito e limitato, il che sicuramente non è.

Alfa - E la seconda ragione?

Omega - E' invece di natura più pratica. Con un Oid nel quale trovano posto le codifiche dei nomi dei produttori di DBMS, quelli stessi valori che avevamo voluto cacciare fuori dalla porta finirebbero col rientrare dalla finestra. Ma c'è di peggio. Un Oid così congegnato ha infatti tutto il sapore dei famigerati "codici parlanti". Cosa succederà al nostro identificatore se un DBMS dovesse per esempio cambiare di mano? Oggi nel mondo dei produttori di database fusioni e acquisizioni sono all'ordine del giorno.

Alfa - C'è poi la terza ragione.

Omega - E' che così si viola apertamente il principio di incapsulazione, un caposaldo del paradigma OO: gli oggetti comunicano tra loro scambiandosi dei messaggi senza doversi preoccupare più che tanto di dove risiedano i loro destinatari, per non parlare poi da quale ODBMS siano gestiti e di chi l'abbia prodotto.

Alfa - Resta il fatto della lunghezza di questi Oid: 96 bit fanno 12 byte, ovvero caratteri. Non sono tantissimi, ma io faccio già fatica a trascrivere senza errori il mio codice fiscale...

Omega - Niente paura. Infatti in un ODBMS che si rispetti difficilmente gli Oid si rendono visibili e quindi gli utenti non sono costretti a usarli per accedere agli oggetti. Infatti quasi tutti gli ODBMS permettono di assegnare agli oggetti dei nomi-utente memorizzati in un dizionario di simboli. Tra l'altro nulla vieta che utenti diversi dispongano ciascuno di un proprio dizionario. Un mito da sfatare

Alfa - E a questo punto avremmo il meglio dei due mondi: degli Oid semanticamente vuoti a garanzia dell'unicità e dell'invarianza della identificazione, dei simboli e cioè dei valori significativi per poter poi usare gli oggetti.

Omega - Per non parlare delle prestazioni! Con gli Oid niente più Join e così le prestazioni migliorerebbero di interi ordini di magnitudine. A quanto scrive il professor Marchesi le interrogazioni sugli oggetti complessi sarebbero "molto più rapide (anche cento o mille volte più rapide) che nei relazionali".

Alfa - Se le cose stanno così, cosa aspetta la comunità informatica a passare armi e bagagli agli ODBMS?

Omega - Non aver tanta fretta. Quello della superiorità in fatto di prestazioni degli ODBMS sugli RDBMS è un mito in parte da sfatare.

Alfa - E chi lo dice?

Omega - Lo dice Won Kim, uno che se ne intende. E' infatti fondatore di UniSQL, oltre che presidente di ACM Sigmod e editor di ACM Transaction on Database System.

Alfa - Sentiamo allora cosa sostiene Won Kim.

Omega - Un esempio potrà forse chiarirci meglio la sua tesi. Nella Figura 1a troviamo due tabelle relazionali; la prima tratta di PERSONE e la seconda di APPARTAMENTI. Anche la Figura 1b tratta di PERSONE e di APPARTAMENTI, ma così come possono essere rappresentati in un ODBMS.

Alfa - Non vedo grandi differenze.

Omega - E invece una differenza c'è. Infatti nella Figura 1a le colonne PERSONA_ID e APPARTAMENTO_ID contengono dei valori assegnati in modo dichiarativo dall'utente, mentre nella Figura 1b gli Oid di PERSONA e APPARTAMENTO sono dei non valori assegnati dal sistema. Immaginiamo a questo punto di voler scoprire dove abita una nostra vecchia conoscenza, quel Signor Rossi che è nato a Roma cinquant'anni fa. Con un ODBMS la ricerca sarà rapidissima: quell'Oid che troviamo nell'oggetto ROSSI è infatti un puntatore a una hash table mantenuta in memoria che contiene a sua volta l'indirizzo fisico del suo APPARTAMENTO. Invece un RDBMS, per riuscire a rin- tracciare l'appartamento del Signor Rossi, dovrà ricorrere a un Join e potrebbe vedersi cosi costretto a scandire sequenzialmente tutta la tabella APPARTAMENTI alla ricerca di un valore di APPARTAMENTO-ID uguale a quel valore di PROPRIETARIO_DI che nella tabella PERSONE indica appunto l'abitazione del Signor Rossi. Bada bene! Le tabelle del nostro esempio sono di due sole righe, ma nelle tabelle relazionali del Mondo Reale le righe si contano a centinaia di migliaia e anche a decine di milioni. Le prestazioni risulterebbero inferiori di interi ordini di magnitudine.

Alfa - Non mi hai convinto. Infatti in ambiente relazionale nessuno si sognerà mai di definire una PK senza costruire su di essa un bel indice. E a questo punto non venirmi a parlare di scansione sequenziale! Le prestazioni del nostro RDBMS saranno del tutto equivalenti a quelle di un ODBMS che usi tecniche di hash table lookup.

Omega - E' vero, le prestazioni saranno le stesse. Ma un ODBMS sa fare di meglio e di più. Può per esempio convertire i riferimenti agli oggetti in puntatori in memoria agli Oid di quelli oggetti. Immaginiamo per esempio di aver caricato in memoria tutti gli oggetti delle Classi PERSONE e APPARTAMENTI; immaginiamo anche che i riferimenti agli

oggetti APPARTAMENTI contenuti negli oggetti PERSONE siano convertiti nell'indirizzo assoluto di questi oggetti. A questo punto si potrà navigare in memoria con un semplice fetch da PERSONA ad APPARTAMENTO senza dover nemmeno passare attraverso una hash table. Sarà la cosa più veloce di questo mondo e i vantaggi in fatto di prestazioni si faranno sentire tanto più quanto maggiore sarà il numero degli oggetti caricati in memoria e il numero degli accessi a questi oggetti, specialmente se gli accessi sono ripetuti. La capacità di navigare attraverso oggetti residenti in memoria è invece una capacità che fa difetto agli RDBMS che nelle loro tuple non memorizzano puntatori ad altre tuple; e i pur ingenti buffer di memoria di cui si sono dotati non permettono di surrogarla. Come ammette lo stesso Won Kim, sotto questo punto di vista "in fatto di prestazioni gli ODBMS possono stracciare gli RDBMS".

## Join addio?

Alfa - Adesso sì che mi hai convinto. La superiorità in fatto di prestazioni degli ODBMS non è solo un mito, del resto lo ammette anche Won Kim. E a questo punto, a quanto mi sembra di capire, Join addio!

Omega - Non essere troppo precipitoso. Dopo questa sua ammissione, ecco infatti cosa scrive Won Kim in un suo recente saggio intitolato "Object-Oriented Database Systems: Promises, Reality, and Future" (in Modern Database Systems - The Object Model, Interoperability, and Beyond, Addison-Wesley, 1995): "Numerose applicazioni che richiedono il lookup degli Oid presentano anche tutta una serie di requisiti di accesso e aggiornamento del database e gli RDBMS sono stati progettati appositamente per soddisfarli. Questi requisiti comprendono il caricamento di massa dei database; la creazione, l'aggiornamento e la cancellazione di singoli oggetti (un oggetto alla volta); il retrieval da una Classe di uno o più oggetti che soddisfanno certe condizioni di ricerca; il join di più Classi; il commit delle transazioni, e via dicendo. Per applicazioni come queste gli ODBMS non hanno particolari vantaggi da offrire".

Alfa - Sarà, ma quanto ai Join non mi hai ancora risposto fino in fondo.

Omega - Resteranno pur sempre necessari. Sono infatti il meccanismo generalizzato che in un RDBMS permette di correlare due relazioni in base ai valori di coppie dei loro attributi. Un ODBMS è in grado in molti casi di sostituire un Join esplicito, così come lo vuole un RDBMS, con un Join implicito. Questo ogni volta che l'Oid di un Oggetto appartenente a una certa Classe viene memorizzato sotto forma di puntatore in un'altra Classe. Ma anche in un ODBMS gli Oid non sono tutto e le Classi possono condividere, e normalmente condividono, coppie di "normali" attributi. Anche per loro i Join restano quindi un qualcosa di utile, per non dire di irrinunciabile. In altre parole un ODBMS che si rispetti dovrà essere in grado di rispondere anche a domande come questa: "Dimmi il nome di tutti i cinquantenni che posseggono un appartamento dalle parti della Stazione Centrale". Qui ci vuole un Join, un semplice fetching degli Oid infatti non basta.

Alfa - E a questo punto eccoci ritornati al punto di partenza. L'identificatore dev'essere un valore o un non valore? Come la pensano al riguardo i signori che stanno mettendo a punto lo standard SQL3?

Omega - I riferimenti agli oggetti avverrebbero in puro spirito OO, e cioè attraverso i membri di un tipo di dato REF basato a sua volta sul valore degli Oid e non delle chiavi.

Quanto poi agli identificatori degli oggetti ho l'impressione che si cerchi di tenere il piede in due, anzi tre scarpe. SQL3 supporterebbe infatti tre sorta di oggetti:

- righe di tabelle prive di Oid e accessibili attraverso la PK come negli attuali RDBMS
- sempre righe di tabelle, ma con un Oid generato dal sistema univoco e invariante nel tempo che si rende visibile come prima colonna di ciascuna relazione; è in buona sostanza quel "surrogato" proposta a suo tempo da Codd
- infine righe di tabelle, ma con un Oid non visibile per gli utenti e non necessariamente univoco e invariante nel tempo.

Alfa - Quest'ultima possibilità mi sembra particolarmente discutibile.

Omega - Tanto discutibile da essere fortemente tentato di seguire il consiglio che ci viene dal Manifesto dei Database della Terza Generazione: "Gli Identificatori Univoci (UID) vanno assegnati ai record dei DBMS solo se non è disponibile una PK definita dall'utente".

**Tecnet Dati s.r.l.**
C.so Svizzera 185 –
10149 - Torino (TO), Italia
Tel.: +39 011 7718090 Fax.: +39 011 7718092
P.I. 05793500017 C.F. 09205650154
www.tecnetdati.com