Domande Orale Turini

Davide Mele

July 23, 2018

- **Clausola ORDER BY**
  ORDER BY ⟨attributi⟩
  ⟨attributi⟩: lista attributi {ASC | DESC}
  Semantica: ordinamenti corrispondenti
  Esempio: ORDER BY R.A, R.B DESC

- **Join naturale**
  Il NATURAL JOIN, date 2 o più tabelle diverse con un attributo avente lo stesso nome, esegue un JOIN in cui l'attributo omonimo compare una sola volta

- **Join in algebra relazionale**
  È l'operatore più tipico dell'algebra relazionale; consente di stabilire connessioni tra dati in relazioni diverse, prendendo come vantaggio la natura "value-based" del modello relazionale. Le versioni principali del JOIN sono 2: il *natural* join, che tiene conto del nome dell'attributo, e il *theta* join, aventi come simbolo $\bowtie$

- **B-Tree, per cosa li abbiamo usati**
  È la struttura dati usata più di frequente nei DBMS. Forniscono accesso associativo (basato su un valore di una chiave, che consiste in uno o più attributi), senza vincoli sulla posizione fisica delle tuple. Sono usati per operazioni di ricerca ed aggiornamento, poiché entrambe le operazioni possono avvenire con complessità logaritmica e con procedure semplici.

- **Group by**
  È l'operatore di raggruppamento. La sintassi è:
  GROUP BY ⟨attributi di raggruppamento⟩
  Semantica: raggruppamento della tabella, divisione in gruppi delle ennuple, raggruppamento sulla base dei valori comuni per gli attributi di raggruppamento

- **Having**
  È l'operatore utilizzato per esprimere condizioni sui gruppi (che coinvolgono funzioni aggregative)

- **Buffer**
  Il Buffer è una grande area di memoria principale pre-allocata al DBMS e condivisa tra le varie transazioni. Il Buffer è organizzato in pagine, di dimensione uguale o multipla ai blocchi input/output usati dal sistema operativo. Le dimensioni delle pagine variano da pochi kbytes a circa un centinaio. I tempi di accesso alla memoria principale sono più veloci di quelli alla memoria secondaria di un ordine di $10\hat{6}$. Può contenere potenzialmente l'intero database.

- **Problema del buffer (rimpiazzamento) e politiche viste**
  Politiche gestione del buffer:
  Politica *Steal*: consente al gestore del buffer di selezionare una pagina allocata ad un'altra transazione, detta "vittima".
  Politica *No-steal*: politica che esclude questa possibilità.
  Politica *Force*: richiede che tutte le pagine attive di una transazione siano scritte nella memoria secondaria prima del consolidamento.
  Politica *No-force*: affida la scrittura delle pagine di una transazione al gestore del buffer.
  Politica *No-steal/No-force*: coppia di politiche preferita dai DBMS.
  C'è anche la possibilità di "anticipare" i tempi di caricamento e scaricamento delle pagine, tramite politiche di *pre-fetching* e *pre-flushing*.

- **Buffer Manager**
  Ha la responsabilità di mantenere temporaneamente porzioni del database in memoria centrale, per aumentare l'efficienza del DBMS e garantire nel contempo l'affidabilità. Il gestore del buffer invia al gestore della memoria secondaria le effettive richieste di lettura e scrittura fisica.

- **Distinct**
  È una keyword posta nella clausola SELECT, precisamente secondo questa sintassi:
  SELECT [DISTINCT] ⟨attributi⟩
  e serve ad eliminare nella proiezione i duplicati dell'attributo che precede.

- **JOIN: cosa fa e complessità costruzione**
  Consente di stabilire connessioni tra dati in relazioni diverse, prendendo come vantaggio la natura "value-based" del modello relazionale. La complessità di un JOIN dipende dalla cardinalità delle tabelle argomento.

- **Concetto selezione, proiezione...**
  Nell'ambiente DB, con "selezione" si intende un sottoinsieme di tuple di una entità che soddisfa una certa condizione; con "proiezione" invece si intende quali attributi di un'entità si vuole che siano inclusi nel result set.

- **Hash table**
  Le strutture hash-based assicurano accessi associativi efficienti ai dati, basati sul valore di un campo chiave, composta da un numero arbitrario di attributi di una tabella data. Una struttura H-B ha B blocchi. Il metodo di accesso utilizza un algoritmo hash, che, una volta applicato alla chiave, restituisce un valore tra 0 e B-1. Questo valore è interpretato come la posizione del blocco nel file, ed usato sia per leggere che per scrivere tuple nel file. È il modo più efficiente di per query con predicati di uguaglianza, ma inefficiente per query con predicati intervallo.

- **Vincolo di tupla**
  Un vincolo di tupla o record è un vincolo che può essere valutato su ciascuna tupla indipendentemente dalle altre. Un vincolo definito con riferimento a singoli valori viene detto vincolo su valori o vincolo di dominio in quanto impone una restrizione sul dominio dell'attributo.

- **Relazioni molti a molti, sia nel modello concettuale che logico (passaggio da uno all'altro)**
  Nel modello logico, un'associazione molti a molti tra due classi si rappresenta aggiungendo allo schema una nuova relazione che contiene due chiavi esterne che riferiscono le due relazioni coinvolte; la chiave primaria di questa relazione è costituita dall'insieme di tutti i suoi attributi. Questa relazione contiene un'ennupla per ogni istanza dell'associazione. Se l'associazione ha degli attributi, questi attributi vengono aggiunti alla nuova relazione, e non vanno a far parte della chiave della nuova relazione

- **Foreign key**
  La Foreign Key, ovvero la chiave esterna, esprime un vincolo d'integrità referenziale tra due o più tabelle. Identifica una o più colonne di una tabella (referenziante) che referenzia una o più colonne di un'altra tabella.

- **Database graph**
  Sono Database di tipo NoSQL con strutturazione opposta a quelli aggregate oriented. Fa uso di nodi e archi per archiviare l'informazione. I record sono di dimensioni ridotte e le interconessioni sono complesse. Rispetto ai DB relazionali, sono più veloci nell'associazione di insiemi dati, e mappano in maniera più diretta le strutture di applicazioni orientate agli

oggetti. Dipendono meno da un rigido schema ER e sono molto più adeguati per gestire dati mutevoli con schemi evolutivi.

- **Definizione modello relazionale (collegato a precedente)**
  Il modello relazionale è un modello logico di rappresentazione o strutturazione dei dati di un database implementato su sistemi di gestione di basi di dati (DBMS), detti perciò sistemi di gestione di basi di dati relazionali (RDBMS). Si basa sulla teoria degli insiemi e sulla logica del primo ordine ed è strutturato intorno al concetto matematico di relazione (detta anche tabella). Per il suo trattamento ci si avvale di strumenti quali il calcolo relazionale e l'algebra relazionale.

- **Join in SQL + Prodotto cartesiano (di questo anche "concetto" matematico sugli insiemi)**
  Il JOIN in SQL tra due o più tabelle si ottiene mettendo la keyword [INNER | NATURAL] JOIN tra le tabelle, nella clausola FROM. Il prodotto cartesiano invece si ottiene con le keyword CROSS JOIN, oppure separando le tabelle con una virgola. Come concetto matematico sugli insiemi, dati due insiemi A e B non vuoti, il prodotto cartesiano A per B è l'insieme che ha per elementi tutte le coppie ordinate (a,b) con a∈A e b∈B.
  $A \times B = \{(a, b) | a \in A \text{ e } b \in B\}$

- **Cardinalità, concetto ed esempi**
  La cardinalità di un'associazione fra X ed Y descrive contemporaneamente la molteplicità dell'associazione della sua inversa.

- **B tree: definizione, differenza B+tree e utilizzo**
  (Vedere sopra per definizione e utilizzo) La differenza è che nei B+tree:
  - le foglie sono collegate da una catena, che le connette nell'ordine imposto dalla chiave
  - supportano efficientemente le query intervallo
  - maggiormente usati dai DBMS

  I B-tree invece:
  - non hanno connessione sequenziale tra le foglie
  - i nodi intermedi usano due puntatori per ciascuna chiave di valore K_i: uno punta direttamente al blocco che contiene la tupla corrispondente a K_i; l'altro punta al sotto-albero con le chiavi maggiori e minori di K_{i+1}

- **Normal Form Boyce Codd**
  Uno schema di relazione $R(X)$ con dipendenze F in forma normale Boyce-Codd (BCNF) se per ogni dipendenza in F^+ della forma $\alpha \to \beta$, con $\alpha \subseteq X$ e $\beta \subseteq Y$, almeno una delle due condizioni vale:

$\alpha \to \beta$, ovvero $\alpha \subseteq \beta$
α è superchiave di $R(X)$

- **Vincolo d'integrità, aka Integrità referenziale**
  Nell'ambito dei RDBMS, l'integrità referenziale è un vincolo di integrità di tipo interrelazionale ovvero una proprietà dei dati che, se soddisfatta, richiede che ogni valore di un attributo (colonna) di una relazione (tabella) esista come valore di un altro attributo in un'altra relazione.

- **Famiglie di colonne**
  È un DB di tipo NoSQL, e il modello di questa classe di database è una struttura aggregata a due livelli. La prima chiave è un row-identifier che seleziona l'aggregato d'interesse. Il secondo livello è un insieme di colonne. Le operazioni consentono di accedere alla riga come un tutt'uno, ma anche di prelevare una particolare colonna

- **Strutture di memorizzazione DB relazionale**
  I DB relazionali hanno a disposizione varie strutture di memorizzazione.
  Sequenziale: caratterizzata da un arrangiamento sequenziale delle tuple nella memoria secondaria. In un'organizzazione *entry-sequenced*, la sequenza di tuple è dettata dal loro ordine d'entrata. Nell'organizzazione ad *array*, le tuple sono arrangiate come in un array, e la loro posizione dipende dai vaolri di un indice (o indici). In un'organizzazione *sequenziale ordinata*, la sequenza delle tuple dipende dal valore assunto da ciascuna tupla dal campo che controlla l'ordine, noto come *campo chiave*

- **Algebra relazionale**
  È una raccolta di operatori che sono definiti sulle relazioni, e producono relazioni come risultati, quindi possono essere combinati per formare espressioni complesse.
  Gli operatori sono: unione, intersezione, differenza, ridenominazione, selezione, proiezione, join.

- **Vincolo NOT NULL**
  È un vincolo di tupla, e impone che il valore dell'attributo in questione non debba essere NULL.

- **Superchiave e chiave**
  Una **superchiave** di una relazione R è costituita da un sottoinsieme di attributi di R tali che quegli attributi da soli bastano ad indicare univocamente le tuple contenute nella relazione. Si dice **chiave**, invece, una superchiave minimale della relazione R. Una superchiave si dice minimale se togliendo anche uno solo degli attributi della superchiave ottengo delle tuple duplicate

- **Gestione del buffer**
  La gestione del buffer avviene per opera del buffer manager, che gestisce le operazioni input/output in risposta alle primitive. Le politiche di gestione del buffer sono simili a quelle di gestione della memoria principale dei sistemi operativi, soggette a:
  Principio di data locality: i dati riferiti correntemente hanno una probabilità più alta di essere riferiti in futuro.
  Legge empirica: l'80% delle applicazioni accede solo al 20% dei dati

- **Transazioni: ACID**
  Con ACID si indica le proprietà che una transazione dovrebbe avere:
  Atomicity (atomicità): rappresenta il fatto che una transazione consiste in un'indivisibile unità d'esecuzione. O tutti gli effetti di una transazione sono resi visibili, o la transazione non ha effetto sul DB.
  Consistency (consistenza): richiede che la transazione non vìoli nessun vincolo d'integrità. I vincoli possono essere verificati immediatamente, durante la transazione (l'operazione che causa la violazione è respinta), o in differita, alla fine della transazione (se un vincolo d'integrità è violato, l'intera transazione viene respinta).
  Isolation (isolamento): richiede che una transazione sia eseguita indipendentemente dall'esecuzione di tutte le altre transazioni concorrenti.
  Durability (persistenza): la persistenza richiede che l'effetto di una transazione che ha eseguito in con successo l'istruzione COMMIT non vada perso (l'effetto dura "per sempre").

- **Traduzione delle associazioni molti a molti**
  Un'associazione molti a molti tra due classi si rappresenta aggiungendo allo schema una nuova relazione che contiene due chiavi esterne che riferiscono le due relazioni coinvolte; la chiave primaria di questa relazione è costituita dall'insieme di tutti i suoi attributi. Questa relazione contiene un'ennupla per ogni istanza dell'associazione. Se l'associazione ha degli attributi, questi attributi vengono aggiunti alla nuova relazione, e non vanno a far parte della chiave della nuova relazione

- **La riga che collega due tabelle cosa è?**
  Rappresenta una dipendenza, o meglio un vincolo d'integrità referenziale (?)

- **NoSQL**
  È un termine usato per indicare la moltitudine di approcci diversi alla gestione di DB. La caratteristiche condivise dai DB NoSQL sono.
  - Non usano il modello relazionale
  - Sono eseguiti efficientemente sui clusters
  - Open-source
  - Costruiti per web-estates del 21° secolo

- Senza schema (schemaless)
I DB abbandonano il modello relazionale delle tabelle e dei riferimenti tramite valori e sono raggruppabili in accordo ai seguenti data model:
- *Aggregate orientation model*: Key-Value, Document, Column-family
- *Graph*

- **Da cosa è formato un database relazionale**
Un DB relazionale è composto da tabelle, in cui ogni colonna è un campo e ogni riga un record. Ciò che rende un database relazionale è la presenza di legami fra le tabelle, di connessioni logiche, ovvero di relazioni.

- **Come esprimeresti l'intersezione dell'algebra relazionale senza utilizzare l'operatore intersezione**
Un modo equivalente a $r1 \bowtie_F r2$ è $(\sigma_F(r1 \bowtie r2))$

- **Document Data Model**
Sono DB NoSQL di tipo aggregate-oriented, consistono ovvero di grandi quantità di aggregati, ciascuno dei quali è dotato di una chiave o un id usato per ottenere i dati. Nel modello Document la struttura dell'aggregato è visibile. Il document deve rispettare vincoli di struttura e di tipo che però offrono maggiora flessibilità di accesso. È possibile effettuare queries basate sui campi dell'aggregato, è possibile recuperare solo parte dell'aggregato e il database può creare indici basati sul contenuto dell'aggregato

- **Definizione di dipendenza funzionale**
Una dipendenza funzionale è un particolare vincolo di integrità semantico per il modello relazionale che descrive legami di tipo funzionale tra gli attributi di una relazione.

- **Definizione di funzione (matematica) e da cosa si differenzia da una relazione**
Le funzioni sono particolari relazioni. Una relazione è una legge di corrispondenza (o associazione) fra elementi di un insieme A e quelli di un insieme B. Per essere una funzione, una relazione deve associare a ogni elemento di A uno e uno solo elemento di B.

- **Definizione di transazione**
Una transazione identifica un'elementare unità di lavoro eseguita da un applicazione, a cui vogliamo allocare particolari caratteristiche di affidabilità e isolamento.

- **Selezione e proiezione: sintassi in SQL e algebra relazionale**
In SQL, la selezione è rappresentata dalla clausola WHERE, con sintassi: WHERE [*condizioni*], in algebra relazionale invece è rappresentato da σ_F con F indicante le condizioni. La proiezione in SQL è rappresentata dalla clausola SELECT, con sintassi SELECT [*TargetList*]; in algebra relazionale è rappresentato da $\pi_Y(r)$

- **BCNF: perché se in forma normale non c'è anomalia di inserimento**
  Non ci sono anomalie poiché, per la forma normale BCNF, i pezzi di informazione indipendente sono separati, uno per relazione.

- **NoSQL (aggregati e grafi)**
  I DB NoSQL possono essere raggruppati in modelli aggregati e grafi. I modelli aggregati consentono strutture complesse come caratteristica primaria. L'aggregato si presenta come l'unità di manipolazione dati e di mantenimento della consistenza. L'organizzazione dei dati in aggregati rende naturale organizzare il calcolo sui cluster, poiché gli aggregati si presentano come l'unità naturale per le duplicazioni e lo sharding. L'organizzazione dei dati in aggregati facilita le applicazioni che possono manipolare aggregati.
  I grafi presentano strutturazione opposta agli aggregati: i record sono di dimensioni ridotte e le interconnessioni complesse. Condividono come modello di base nodi e archi. Offrono la navigazione sulle relazioni come operazione di base poco costosa.

- **Perché bisogna fare una tabella aggiuntiva con una relazione molti a molti?**
  Poiché dal punto di vista computazionale, conviene aggiungere una tabella che tenga conto solo delle associazioni tra le chiavi, dato che la cardinalità indica un'associazione di tuple multiple per ogni tupla di una delle due tabelle

- **Cosa fa l'ottimizzatore?**
  L'ottimizzatore decide la strategia migliore di accesso ai dati, per garantire la risposta query più veloce. Riceve una query, dalla quale esegue un'analisi lessicale, sintattica e semantica, per identificare possibili errori. Poi trasforma le query corrette in una forma interna e sceglie la strategia migliore di accesso ai dati.

- **Operatore ridenominazione in algebra relazionale**
  Sia r una relazione definita su un set di attributi X e sia Y un altro set di attributi con cardinalità medesima. Inoltre, siano $A_1A_2...Ak$ e $B_1B_2...Bk$ rispettivamente un ordine di attributi in X e Y. La ridenominazione
  $\rho_{B_1B_2...B_k} \leftarrow_{A_1A2...A_k} (r)$
  contiene una tupla t' per ciascuna tupla t, definita come: t' è una tupla di Y e $t'[B_i] = t[A_i]$ per $i = 1...n$

- **Proprietà algebriche della JOIN**
  Il JOIN (naturale) è sia commutativo che associativo; infatti:
  $E_1 \bowtie E_2 \equiv E_2 \bowtie E_1$ (commutativo)
  $(E_1 \bowtie E_2) \bowtie E_3 \equiv E_1 \bowtie (E_2 \bowtie E_3) \equiv E_1 \bowtie E_2 \bowtie E_3$ (associativo)

- **JSON: descrizione generale**
  JSON è un semplice formato per lo scambio di dati, indipendente dal linguaggio di programmazione, basato su un sottoinsieme di JavaScript, facile da capire, manipolare e generare. Inoltre può essere analizzato nativamente in JS usando *eval()*. Ha in comune con XML il formato di solo testo; è "self-describing" e gerarchico, ma a differenza di quest'ultimo, è più leggero e veloce; JSON usa oggetti tipizzati; meno sintassi, niente semantica; le proprietà sono immediatamente accessibili a JS. Tuttavia, manca di spazi di nomi, non c'è la validazione dell'ereditarietà e non è estendibile

- **Indici**
  Un indice su un attributo A di una relazione R è un insieme ordinato di coppie $(k_i, \{r_j\}_i)$, dove k_i è un valore di A presente in un record di R, ed $\{r_j\}_i$ è l'insieme dei riferimenti ai record di R in cui A vale k_i. In genere un indice è organizzato a B+Tree per permettere di trovare con pochi accessi, a partire da un valore v, i record di R in cui il valore di A è una relazione specificata con v. Un indice può anche essere definito su un insieme di $A_1...A_n$ attributi.

- **Strutture sequenziali**
  Le strutture sequenziali sono strutture dati usate nei DB relazionali. Sono caratterizzate da una disposizione sequenziale di tuple nella memoria secondaria. Il file è composto da vari blocchi di memoria, e le tuple sono inserite nei blocchi secondo una sequenza. A seconda dell'applicazione, la sequenza può essere: *entry-sequenced*, *array*, *sequentially ordered*.