Analisi dei vantaggi dell'adozione di modelli dati XML per un'applicazione di marketplace

Scenario

L'applicazione da realizzare può essere vista come un caso particolare del classico Marketplace. Ovviamente si tratta di un'applicazione web e va realizzata in tecnologia Java (1.6) con database PostgreSQL (da stabilire se 8 o 9) su server Linux.

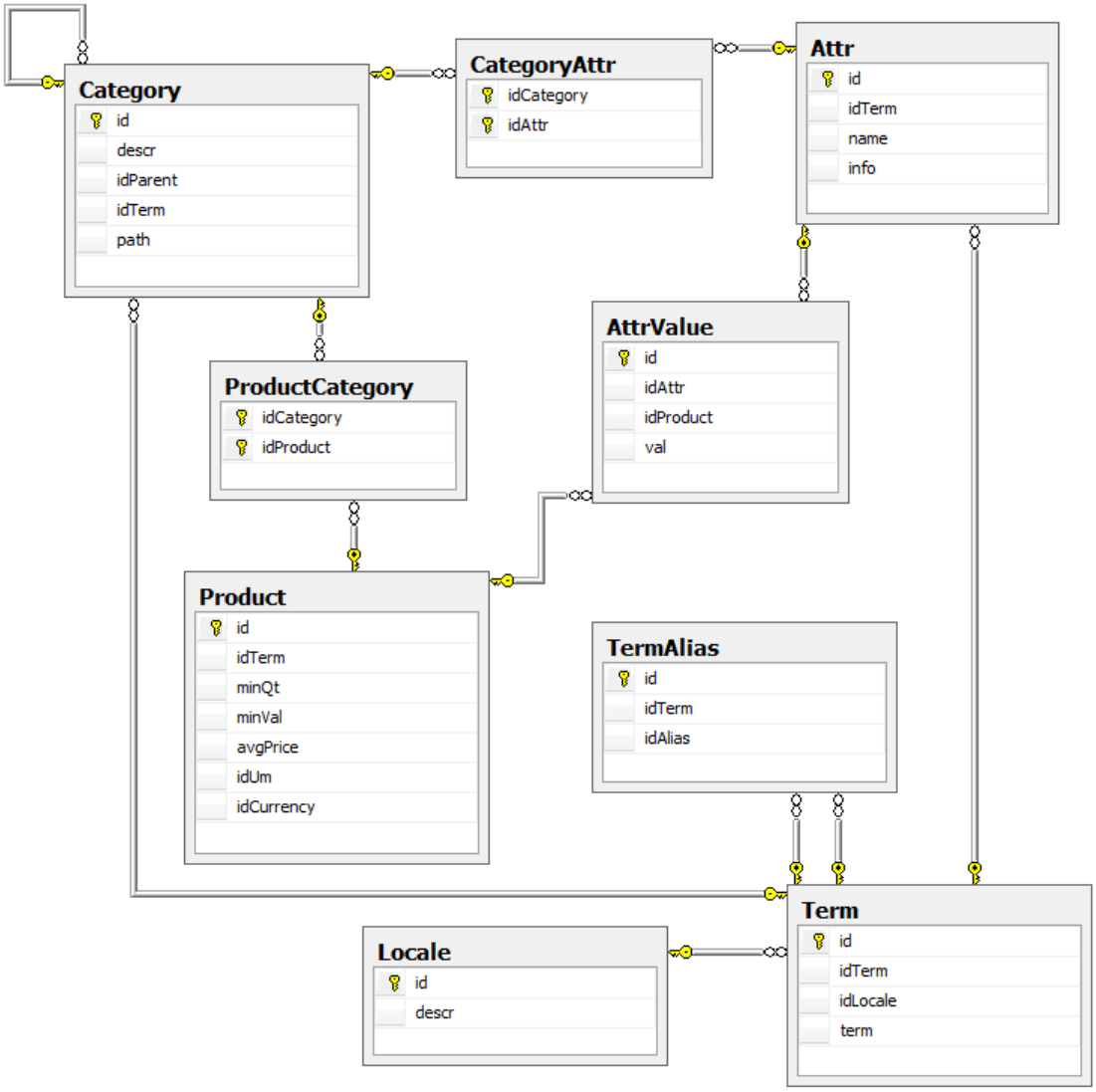
Realtà d'interesse

Nel marketplace entrano delle *aziende,* tramite una normale procedura di registrazione, ciascuna delle quali è interessata ad operare in una particolare *area geografica*, che ha un proprio *catalogo merci*. Le merci vengono gestite centralmente, nel senso che il catalogo delle merci è unico per tutto il sistema, e ogni azienda deve inserire i propri prodotti all'interno di esso. Quindi ad esempio il sistema deve essere in grado di stabilire quali aziende vendono uno stesso prodotto, e non devono essere create istanze diverse di uno stesso prodotto solo a causa di una nomenclatura leggermente diversa (la nomenclatura diversa, ove sia dovuta a utilizzo di sinonimi, va comunque rispettata. Vale lo stesso discorso che si fa nel seguito per le traduzioni). Ciò ovviamente comporta del lavoro manuale al quale è preposta una redazione centrale.
Ogni azienda ha poi una propria "lingua" di riferimento, che è quella che utilizza, ad esempio, per descrivere i suoi prodotti. L'archivio dei prodotti del sistema deve prescindere dalla lingua utilizzata, e quindi deve disporre delle traduzioni del caso per garantire, anche in questo caso, che lo stesso prodotto col nome in lingua A sia lo stesso indicato da una seconda azienda in lingua B.
I prodotti vengono inseriti in un albero di categorie di profondità arbitraria, ed anche per le categorie vanno gestite opportunamente le varie traduzioni.
Ogni azienda ha a catalogo un certo numero di *marche*, e i prodotti possono appartenere a una di queste.
Ogni prodotto è inoltre caratterizzato da *attributi* variabili in base alla categoria di appartenenza; ad esempio un prodotto appartenente alla categoria "personal computer" avrà degli attributi come "processore", "ram", "disco" etc., che verranno anche ereditati dalle sottocategorie (ad es. la sottocategoria "notebook" avrà quegli attributi più altri ad essa specifici).
Inoltre per il prodotto viene specificato un prezzo medio (in una determinata *divisa*), una *unità di misura,* una quantità minima di vendita (a cui si riferisce il prezzo medio).
Per le unità di misura e per le divise vanno gestite le conversioni.
L'utente del sistema dovrà essere in grado di effettuare interrogazioni utilizzando i termini nella propria lingua, potendo specificare sia le aree geografiche d'interesse, sia le categorie e i valori di eventuali attributi specifici della categoria (e delle sopra-categorie) sia range di prezzo e

quantità.

Modellazione standard

La modellazione della realtà sopra descritta in termini di tabelle in un database relazionale classico presenta alcuni punti critici; una possibilità (teorica) è rappresentata nella figura seguente:



Per ora non consideriamo le tabelle *Locale, Term e TermAlias*, che analizzeremo dopo, e trascuriamo anche la presenza del campo *idTerm* nelle altre tabelle. In questo modo la struttura dovrebbe essere chiara: le categorie (*Category*) sono gerarchiche (*idParent* referenzia la chiave della tabella ProductCategory stessa, definendo un albero) e legate agli attributi (*Attr*). Gli attributi hanno un valore quando sono legati ad un prodotto (tramite *AttrValue*). Un prodotto può appartenere a una o più categorie (*ProductCategory*). Per ora trascuriamo discorsi relativi alla tipologia degli attributi o alla profondità dell'albero (si tratta di un esempio teorico più che altro). Eseguire query su questo schema (ad es. trovare tutti i pc con processore i3 sotto i 2 chili di

peso) porta alla costruzione di una query davvero pesante per il dbms (vedi oltre).
La questione delle traduzioni ci porta ad un problema analogo: nella figura viene definita una tabella *Locale*, contenente il contesto linguistico di riferimento. Ogni termine, che è poi ad esempio il nome di una categoria o di un attributo per quel particolare contesto linguistico, ha le varie traduzioni nella tabella *Term* (campo term). Ogni termine ha anche uno o più sinonimi (*TermAlias*: idAlias referenzia idTerm in Term).
L'utente ovviamente specifica la ricerca in una lingua di sua scelta, e il sistema deve rispondere offrendo la traduzione opportuna dei risultati.

Altre ipotesi di modellazione

Tornando alla discussione sulla modellazione degli attributi, una possibilità è sfruttare le funzionalità offerte dal dbms relative alla gestione dei campi XML. Gli attributi (e i relativi valori) dei prodotti potrebbero essere rappresentati da un campo XML inserito direttamente nella tabella *Product*. Una cosa analoga potrebbe essere fatta per la definizione degli attributi legati ad una categoria. Ciò risparmierebbe la faticosa modellazione delle tabelle *Attr* e *AttrValue*, e probabilmente le ricerche potrebbero beneficiarne allo stesso modo.
Viene spontaneo tentare di estendere questo discorso anche alle categorie: il modello di tabella "ricorsiva" è molto compatto ma pesante da interrogare.
Potrebbe inoltre essere ragionevole utilizzare entrambi gli approcci, cioè mantenere alcune definizioni di tabelle presentate così come sono nella figura affiancando ad esse opportune strutture XML per consentire query più efficienti.

Obiettivo dello studio

L'obiettivo dello studio richiesto è proprio quello di individuare modelli della realtà esaminata (a partire dal modello relazionale "puro" e spostandosi verso quello "basato su XML") confrontando pregi e difetti delle soluzioni, mirando in ogni caso alla soluzione del problema in un ambiente reale. E' necessario fornire sia gli schemi del database che confrontare le prestazioni relative alle query più comuni eseguite sul sistema.
A questo scopo precisiamo lo scenario di test:

- la piattaforma è PostgreSQL 8.3 su linux
- le categorie sono organizzate in un albero con 30 elementi al primo livello, 10 al secondo livello, 10 al terzo (ovviamente non perfettamente bilanciato: inoltre non tutte le categorie hanno il terzo livello di profondità, qualcuna può avere il quarto).
- gli attributi per categoria mediamente sono dieci ma possono essere molti di più
- le lingue gestite dal sistema sono 10
- i prodotti sono 500.000
- le query d'interesse sono in primo luogo la ricerca di prodotti con filtro a piacere sugli attributi (ove ha senso per range di valori o *like*), e ovviamente quelle per l'aggiornamento dell'archivio dei prodotti, delle categorie e degli attributi.