Il Rapporto di Ereditarietà in SQL

Il rapporto di ereditarietà rappresenta in realtà un rapporto di specializzazione. Un oggetto di tipo A può anche essere di tipo B, dove B costituisce una sottocategoria di A. Consideriamo il caso in cui vogliamo tracciare, esclusivamente per gli insegnanti, informazioni come la scuola di riferimento (denominata "scuola polo"), gli anni di anzianità e il titolo di studio, oltre alle materie per cui sono abilitati.

Una prima soluzione consisterebbe nel modificare la tabella Person aggiungendo le colonne relative ad anzianità, titolo di studio e scuola polo. Tuttavia questo approccio presenterebbe un inconveniente significativo: tali colonne resterebbero vuote per tutte le righe non relative a insegnanti. L'obiettivo è invece simulare l'ereditarietà, che non esiste nei sistemi di gestione di database SQL standard, attraverso un rapporto uno a uno opzionale. L**'approccio prevede la creazione di una tabella "figlio" contenente i dati del sottotipo, separati da quelli del supertipo che faranno invece capo a una tabella "padre".**

La Struttura della Tabella Teacher

La categoria più generale, rappresentante la classe "padre", viene mappata sulla tabella Person vista in precedenza. A questa si aggiunge una tabella Teacher con la seguente struttura:

```
sqlCREATE TABLE TEACHER (
    ID INT PRIMARY KEY,
    SCHOOL VARCHAR(100),
    DEGREE VARCHAR(100),
    SUBJECTS VARCHAR(100),
    YEARS INT,
    FOREIGN KEY(ID) REFERENCES PERSON(ID) ON CASCADE UPDATE ON DELETE RESTRICT
);
```

La colonna ID in Teacher svolge una duplice funzione. Internamente a Teacher rappresenta la chiave primaria, ma costituisce anche una chiave esterna verso Person. Questo significa che una riga di Teacher deve necessariamente avere una riga di Person di riferimento per essere completa. Teacher non conosce autonomamente il proprio nome, la propria città o la propria data di nascita, informazioni che risiedono in Person.

Di contro, *Person non è vincolata ad avere chiavi esterne, poiché per Person questa relazione non è obbligatoria ma opzionale.* Una riga di Person potrebbe essere collegata a una riga di Teacher, ma potrebbe anche rappresentare una persona che svolge un altro lavoro, nel qual caso non troveremo l'ID di quella Person nella tabella Teacher. Il rapporto è obbligatorio per Teacher, dato che tutti gli insegnanti sono persone, ma opzionale per Person, poiché non tutte le persone sono insegnanti.

> Esempio Pratico di Relazione

> Considerando due tabelle semplificate dove Person contiene gli ID 1, 2 e 3 con i nomi A, B e C rispettivamente, mentre Teacher contiene solo gli ID 1 e 3 con le scuole S1 e S2, possiamo osservare un aspetto importante: l'ID 2 manca in Teacher. Questa assenza è significativa e spiega perché l'ID di Teacher non è definito come auto_increment. Potremmo dover saltare dei numeri perché l'ID di Teacher non ha esistenza autonoma, ma deve necessariamente essere uguale a un ID esistente in Person.

> Nell'esempio, i signori A e C sono insegnanti, quindi le righe 1 e 3 di Person sono collegate con righe corrispondenti in Teacher, e lavorano rispettivamente in S1 e S2. Il signor B si occupa di altro, e dai dati disponibili non sappiamo di cosa. Potrebbe essere uno studente, ma non abbiamo informazioni a riguardo.

Il Predicato di INNER JOIN

Poste due tabelle con un rapporto uno a uno, che è sempre opzionale per la tabella "padre" rappresentante la superclasse, il predicato di INNER JOIN si ottiene semplicemente uguagliando le chiavi primarie. La forma generale è TabellaPadre.PK uguale a TabellaFiglio.PK, o più correttamente TabellaSuperClasse.PK uguale a TabellaSottoClasse.PK. Solo in questo caso specifico è sufficiente uguagliare gli ID: Person.ID uguale a Teacher.ID.

L'INNER JOIN può essere scritto come `SELECT * FROM PERSON, TEACHER WHERE PERSON.ID = TEACHER.ID` e produrrà un risultato contenente solo le righe dove esiste corrispondenza tra Person e Teacher. Nell'esempio precedente, otterremmo le righe relative ad A con scuola S1 e C con scuola S2, mentre B verrebbe escluso dal risultato.

La Forma Esplicita di INNER JOIN

SQL dispone di una forma esplicita con un operatore apposito, INNER JOIN, che permette di imporre il predicato di join fuori dalla clausola WHERE. Questo approccio consente di separare adeguatamente i predicati di join, che rappresentano condizioni strutturali, dalle condizioni di ricerca dei dati, che costituiscono condizioni di dominio.

La forma esplicita per il nostro INNER JOIN, da preferire rispetto alla forma implicita vista finora, è `SELECT * FROM PERSON INNER JOIN TEACHER ON PERSON.ID = TEACHER.ID`. La forma generale è `SELECT * FROM T1 INNER JOIN T2 ON (PREDICATO DI JOIN)`, e questa è la forma che useremo sempre d'ora in avanti. Il risultato rimane identico a quello calcolato in precedenza, si tratta semplicemente di una forma più elegante.

Il Significato di INNER

La parola chiave INNER indica che dobbiamo prendere solo le righe per cui la condizione specificata è vera, vale a dire per cui vale la relazione. Possiamo ricordare questo concetto come "prendi le righe collegate e solo quelle". Applicando l'INNER JOIN ai dati di esempio, perdiamo le informazioni relative al signor B, poiché la seconda riga di Person con ID 2 e nome B non ha corrispettivi nella tabella Teacher. Non partecipa alla relazione e quindi viene esclusa dal risultato dell'INNER JOIN.

Il LEFT JOIN come Alternativa

Potremmo voler visualizzare anche B nel risultato. Chiaramente non potremmo associargli una scuola, non essendo lui un insegnante, ma potremmo volerlo vedere nella lista comunque. Per includere anche B possiamo utilizzare un join diverso, tipicamente usato per queste eventualità, che prende il nome di LEFT JOIN. La sintassi generale è `SELECT * FROM T1 LEFT JOIN T2 ON (PREDICATO DI JOIN)`.

Questa istruzione si legge come segue: prendi le righe di T1 e collegale alle righe di T2 per cui vale la relazione espressa dal predicato di join. Se non trovi righe di T2, presenta comunque la riga di T1 e combinala con una riga di valori NULL per i campi di T2. Possiamo ricordare questo concetto come "preserva tutte le righe di T1" o "preserva tutte le righe a sinistra", che partecipino o meno alla relazione.

Applicando `SELECT * FROM PERSON LEFT JOIN TEACHER ON PERSON.ID = TEACHER.ID` ai nostri dati, otterremo tutte e tre le righe: A con scuola S1, B con valori NULL per i campi di Teacher, e C con scuola S2. La riga 2, il signor B che non partecipa alla relazione con Teacher, è comunque presente ed è stata associata a una riga composta di valori NULL.

Considerazioni sul LEFT JOIN

Il LEFT JOIN non escluderà mai le righe a sinistra, che possono essere quelle della tabella rappresentante la superclasse, ma possono anche essere quelle della tabella lato uno in un rapporto uno a molti, come vedremo in seguito. È importante notare che l'ordine conta: è la tabella che compare a sinistra dell'operatore LEFT JOIN che viene preservata. Non ci sono garanzie di preservare quella a destra.
