Il linguaggio SQL
L'istruzione CREATE TABLE

http://www.vbsimple.fbi/database/db_08_11.htm

Difficoltà: 3 / 5

La prima istruzione DDL è sicuramente relativa alla creazione delle tabelle: CREATE TABLE consente di creare una nuova tabella specificando uno o più campi accompagnati dal rispettivo tipo di dato. Taluni database consentono anche di specificare la presenza di uno o più indici o chiavi per la tabella da creare. La sintassi generale dell'istruzione è la seguente:

```
CREATE TABLE <tabella> (
  <campo> <tipo> [ NULL | NOT NULL ] [ <chiave> ] [ <vincolo> ],
  <campo> <tipo> [ NULL | NOT NULL ] [ <chiave> ] [ <vincolo> ],
  ...
)
```

Nella creazione di una tabella è necessario indicare per ciascun campo il nome ed il tipo che si intende assegnare; opzionalmente potranno essere indicati alcuni vincoli per ciascun campo. Il primo vincolo riguarda naturalmente il tipo di dato: un campo definito numericamente non potrà naturalmente contenere caratteri alfanumerici e così un campo data non potrà contenere altro che date; si tratta di un vincolo implicito e legato al tipo di campo stesso.

A questo primo vincolo è possibile assegnarne altri come quello che riguarda il valore NULL. Sarà possibile ammettere o negare la presenza di tale valore utilizzando la clausola NULL oppure NOT NULL. Se fosse negata la possibilità di inserire un valore NULL all'interno di un campo e venisse comunque effettuato l'inserimento di tale valore esplicitamente oppure durante un inserimento non venisse specificato un valore per il campo (ed in assenza di un valore di default venga assunto NULL), sarà generato un errore.

Non è stata inclusa una tabella con i nomi dei tipi di campo poiché essi dipendono dalla combinazione di database e driver per accedere allo stesso; sebbene esistano tipi di campi comuni come il numero intero, ciascun database utilizza la propria terminologia e ciascun driver per accedere ad un database utilizza un proprio dialetto. Sarà compito del driver stesso riconvertire la richiesta nel tipo di dato utilizzato internamente al database.
Nella creazione di una tabella sarà necessario specificare almeno un campo ed il relativo tipo di dato; non è infatti possibile creare tabelle senza campi.

A questi due vincoli si associa il concetto di chiave (quasi sempre unita alla definizione di un indice per essa); un solo campo della tabella può essere chiave primaria (Primary key), mentre più campi possono essere chiavi esterne (Foreign key) di altre tabelle. Tuttavia molti campi rappresentano dati separati e non entreranno in nessuna delle due chiavi.

Alcuni database consentono di affiancare un ulteriore vincolo specifico formato solitamente dalla parola chiave CHECK o CONSTRAINT e da un'espressione che indica la condizione che ogni record deve soddisfare per poter essere ospitato all'interno della tabella.

Per una spiegazione dei concetti di tipo di campo, indice e chiave si rimanda all'Introduzione ai Database.

Questa lunga premessa sarebbe inutile se non fosse accompagnata da una serie di esempi esplicativi. Quelli che seguiranno sono basati su un **meta linguaggio** standard ma con molta probabilità necessiteranno di piccoli accorgimenti per renderli funzionanti sui singoli database.

```
CREATE TABLE dipendenti (
  id         INTEGER,
  nome       CHAR(50),
  cognome    CHAR(50),
  ruolo      CHAR(20),
  assunzione DATETIME
)
```

Questo esempio specifica solamente i dati minimi necessari: il nome della tabella **dipendenti**, composta da 5 campi di cui solo il primo è numerico, i successivi 3 alfanumerici e l'ultimo di tipo data/ora. Il valore racchiuso tra parentesi rappresenta un parametro di creazione per il tipo di dati che lo precede. *CHAR(50)* indicherà al database che il campo sarà di tipo carattere e con una lunghezza (se massima oppure fissa dipende dal singolo database) di 50 caratteri. Il valore 50 rappresenta naturalmente il parametro di creazione richiesto dal tipo di dato *CHAR*. Altri tipi di dati non richiedono tale parametro (vedi ad esempio *INTEGER* o *DATETIME*), mentre altri ne richiedono più di uno (ad esempio *DECIMAL* richiederebbe due parametri, uno per la lunghezza ed uno per i decimali, denominata scala).

In questo esempio non sono stati specificati vincoli ed il database assumerà le opzioni predefinite; per qualche database questo significherà di proibire l'uso dei valori NULL mentre per qualche altro significherà che sarà possibile inserire valori NULL.

```
CREATE TABLE dipendenti (
  id         INTEGER  NOT NULL,
  nome       CHAR(50) NOT NULL,
  cognome    CHAR(50) NOT NULL,
  ruolo      CHAR(20) NULL,
  assunzione DATETIME NOT NULL
)
```

Questo esempio analogo al precedente si differenzia soltanto per la specifica esplicita dei valori NULL; soltanto il campo **ruolo** potrà contenere valore NULL. Questo significa che durante un inserimento (vedi Istruzione INSERT INTO) dovrà essere fornito un valore esplicito per i campi **id**, **nome**, **cognome** e **assunzione**; sarà invece possibile evitare di specificare un valore per il campo **ruolo** che assumerà quindi il valore NULL ammesso.

Affrontiamo la problematica della specifica delle chiavi nella creazione di una tabella: come già detto le chiavi possono essere di tipo primario o di tipo esterno; le prime indicano valori che verranno referenziati in altre tabelle, le seconde referenziano invece campi di altre tabelle. Una chiave primaria implica nella quasi totalità dei casi l'univocità della riga all'interno dell'intera tabella; non sarà quindi possibile avere righe con quel determinato campo duplicato e altresì non sarà possibile inserire valore NULL in quel campo.

Per taluni database basterà specificare la parola chiave PRIMARY KEY (oppure soltanto PRIMARY) a fianco del tipo di dati del campo cui si riferisce, per gli altri database o per avere chiavi primarie multicampo, invece, è necessario indicare una clausola apposita per la creazione del vincolo della chiave primaria. Vediamo entrambi i casi:


```
CREATE TABLE dipendenti (
  id         INTEGER  NOT NULL PRIMARY KEY,
  nome       CHAR(50) NOT NULL,
  cognome    CHAR(50) NOT NULL,
  ruolo      CHAR(20) NULL,
  assunzione DATETIME NOT NULL
)
```

Questa prima definizione eseguirà la creazione della tabella **dipendenti** ed assegnerà la chiave primaria al campo **id**. Si tratta di una chiave primaria legata ad un solo campo; per determinate necessità o situazioni è necessario operare come segue:

```
CREATE TABLE <tabella> (
  <campo> <tipo> [ NULL | NOT NULL ],
  <campo> <tipo> [ NULL | NOT NULL ],
  ...
  CONSTRAINT <vincolo> PRIMARY KEY (<campi chiave>)
)
```

La seguente forma può cambiare da un database all'altro per cui si rimanda alla guida del singolo database; per certuni il nome del vincolo deve essere uguale al nome del campo che vincolano, per altri invece deve essere un nome univoco all'interno di tutto il database (quindi due tabelle non possono avere lo stesso nome del vincolo). Dato l'esempio precedente, questa nuova forma si espone con:

```
CREATE TABLE dipendenti (
  id         INTEGER  NOT NULL,
  nome       CHAR(50) NOT NULL,
  cognome    CHAR(50) NOT NULL,
  ruolo      CHAR(20) NULL,
  assunzione DATETIME NOT NULL,
  CONSTRAINT pk_dipendenti_id PRIMARY KEY (id)
)
```

A completare l'utilizzo delle chiavi primarie vi è la creazione delle chiavi esterne (FOREIGN KEY) per una tabella che referenzia una chiave primaria di un'altra tabella. La sua forma è:

```
CREATE TABLE <tabella> (
  <campo> <tipo> [ NULL | NOT NULL ],
  <campo> <tipo> [ NULL | NOT NULL ],
  ...
  CONSTRAINT <vincolo> FOREIGN KEY (<campi chiave>)
  REFERENCES <tabella 2> (<campi chiave 2>)
)
```

Ricordiamo che l'utilità di una chiave esterna è quella di poter effettuare una relazione di dipendenza tra i valori di una tabella ed un'altra. In tal modo non sarà infatti possibile inserire valori in una tabella dipendente senza che vi sia una corrispondenza con la tabella relazionata. Riprendendo l'esempio precedente della tabella **dipendenti** con chiave primaria sul campo **id**, è possibile stabilire una relazione di riferimento ad una nuova tabella di nome **paghe** come segue:

```
CREATE TABLE paghe (
  numero     INTEGER  NOT NULL,
  stipendio  NUMERIC(10,2) NOT NULL,
  CONSTRAINT fk_paghe_id FOREIGN KEY (numero)
  REFERENCES dipendenti (id)
)
```


Il campo id della tabella **paghe** sarà legato al campo id della tabella **dipendenti**; i nomi dei campi referenziati sono stati volutamente differenziati per chiarire il funzionamento: la tabella referenziata (dipendenti) possiede una chiave primaria sul campo **id**, mentre la tabella dipendente (paghe) possiede una chiave esterna sul campo **numero** che punta al campo **id** della prima tabella.

In tal modo se dovesse essere tentato l'inserimento di una riga nella tabella paghe con il valore nel campo numero inesistente nel campo id della tabella dipendenti sarà generato un errore. Questo concetto di controlli incrociati tra campi di tabelle (ovvero tra le loro chiavi) è detto **integrità referenziale**.

Affinché sia possibile effettuare la relazione è necessario che la tabella referenziata contenga una chiave primaria oppure un *indice unico* su quel campo. La creazione di un indice unico è similare alla creazione di una chiave primaria, ma a differenza di questa è possibile avere più di un indice unico. La sua definizione è la seguente:

```
CREATE TABLE <tabella> (
  <campo> <tipo> [ NULL | NOT NULL ],
  <campo> <tipo> [ NULL | NOT NULL ],
  ...
  CONSTRAINT <vincolo> UNIQUE (<campi chiave>)
)
```

Il suo utilizzo è uguale a quello della chiave primaria e pertanto non saranno mostrati esempi del caso.

Citiamo solo per completezza la presenza di ulteriori clausole quali **DEFAULT** che consente di assegnare un valore predefinito ad un campo se questo non viene specificato durante l'operazione di inserimento. In assenza di tale specifica il campo assumerà valore NULL; **IDENTITY** identifica un campo con un valore contatore autoincrementato; CHECK consente invece di effettuare determinati controlli sui valori inseribili all'interno del campo controllato, ad esempio verificare che il valore del campo sia un valido indirizzo di posta elettronica.

Tali clausole sono specifiche per alcuni database e la loro sintassi varia totalmente da un database all'altro e pertanto non saranno affrontati in questa sede.

Fibia FBI
5 Maggio 2004

Torna all'indice della sezione Database
