Aspetti relazionali ad oggetti di SQL:1999

Utilizzo di un DBMS

- Tre classi di applicazioni:
  - applicazioni gestionali
  - applicazioni navigazionali
  - applicazioni multimediali

ORDBMS: caratteristiche generali

- Nuovi tipi di dato
  - testi, immagini, audio/video, dati geografici, ecc.
  - tipi di dato user-defined
  - metodi per modellare le operazioni sui tipi definiti dall'utente
- Interrogazioni per contenuto su dati multimediali
- Trigger & stored procedure

Estensione del sistema di tipi

- Tipi semplici
- Tipi complessi:
  - ADT
  - row types
- Tipi riferimento
- Tipi collezione

Tipi semplici

- I tipi semplici (o distinct type) sono la forma più semplice di estensione del sistema dei tipi fornita da un ORDBMS
- Consentono agli utenti di creare nuovi tipi di dati, basati su un solo tipo primitivo
- Sono usati per definire tipi di dati che richiedono operazioni diverse rispetto al tipo primitivo su cui sono definiti

Tipi complessi

- Un tipo complesso, o strutturato,include uno o più attributi
- Analoghi alle struct del C o ai record del Pascal
- possono essere usati come:
  - tipi di una colonna in una relazione (ADT)
  - tipi di una tabella (row types)

Abstract data types

```
CREATE TYPE t_indirizzo(
    numero_civico     INTEGER,
    via               VARCHAR(50),
    città         CHAR(20),
    stato         CHAR(2),
    cap               INTEGER);
```

t_indirizzo è un tipo complesso i cui attributi hanno tipi predefinitii

Abstract data types

- Gli ADT possono essere usati come tipi di una colonna di una relazione:

```
CREATE TABLE Impiegati (
        imp#        id_impiegato,
        nome        CHAR(20),
        curriculum TEXT,
        indirizzo   t_indirizzo);
```

Abstract data types

- Gli ADT possono anche essere annidati
- Sugli ADT possono essere definiti metodi come parte della definizione del tipo
- Gli ADT sono totalmente incapsulati
- La loro manipolazione può avvenire solo mediante apposite funzioni automaticamente create dal DBMS al momento della creazione dell'ADT

Row types

- Un tipo complesso può anche essere usato come tipo di una intera tabella (row type)
- Le righe della tabella sono istanze del tipo complesso mentre le colonne coincidono con gli attributi del tipo complesso

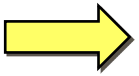
Row types

- Permettono di definire un insieme di tabelle che condividono la stessa struttura
- Permettono di modellare in modo intuitivo le associazioni tra dati in tabelle diverse
- Consentono di definire gerarchie di tabelle

Esempio (relazionale)

Impiegati

imp#		prj#
SM123	...	12



Progetti

prj#	nome	
12	Oracle	....

Row types

- In un ORDBMS ho due opzioni in più:
  - definire un ADT t_progetto e usare questo come tipo di una colonna della relazione Impiegati (ridondanza dei dati perché lo stesso progetto può essere memorizzato molte volte in Impiegati)
  - definire una tabella basata su un nuovo tipo complesso e riferire le colonne istanza di questo nuovo tipo

Esempio

```
CREATE TYPE t_progetto(
   prj#   INTEGER,
   nome VARCHAR(20),
   descrizione VARCHAR(50),
   budget       INTEGER);

CREATE TABLE Progetti OF t_progetto;
```

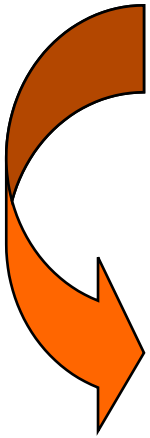
Tipi riferimento

- I row type possono essere combinati con i tipi riferimento (REF type)
- Permettono di rappresentare facilmente le associazioni tra istanze di tipi
- Tali tipi permettono ad una colonna di riferire una tupla in un'altra relazione
- Una tupla in una relazione viene identificata tramite il suo OID

Esempio

```
CREATE TABLE Progetti OF t_progetto
 (prog_ref                REF(t_progetto));
```

prog_ref	prj#	nome	descrizione	budget
xxxx	433	DB2	ORDBMS	20,000,000



Riferimento univoco creato automaticamente dal DBMS

Esempio

```
CREATE TABLE Impiegati(
   imp# id_impiegato,
   nome VARCHAR(50),
   indirizzo      t_indirizzo,
   assegnamento      REF(t_progetto) SCOPE Progetti);
```

Più impiegati possono riferire lo stesso progetto

Un impiegato è assegnato al massimo ad un progetto

Tipi collezione

- Set
- Bag
- List
- Array

Esempio

```
CREATE TABLE Impiegati(
   imp#            id_impiegato,
   nome            VARCHAR(50),
   competenze                  SET(VARCHAR(20)),
   titolo_di_studio    VARCHAR(30));
```

Esempio

```
CREATE TYPE t_impiegato(
    imp# id_impiegato,
    nome VARCHAR(30),
   indirizzo      t_indirizzo,
    manager      REF(t_impiegato),
    progetti      SET(REF(t_progetto)),
    figli    LIST(REF(t_persona)),
    hobby         SET(VARCHAR(20)));

CREATE TABLE Impiegati2 OF t_impiegato;
```

Ereditarietà

- Possibilità di definire relazioni di supertipo/sottotipo
- L'ereditarietà consente di specializzare i tipi esistenti a seconda delle esigenze dell'applicazione
- Un sottotipo eredita gli attributi e i metodi dei suoi supertipi

Ereditarietà

- Si possono distinguere due tipi di ereditarietà
  - ereditarietà di tipi
  - ereditarietà di tabelle

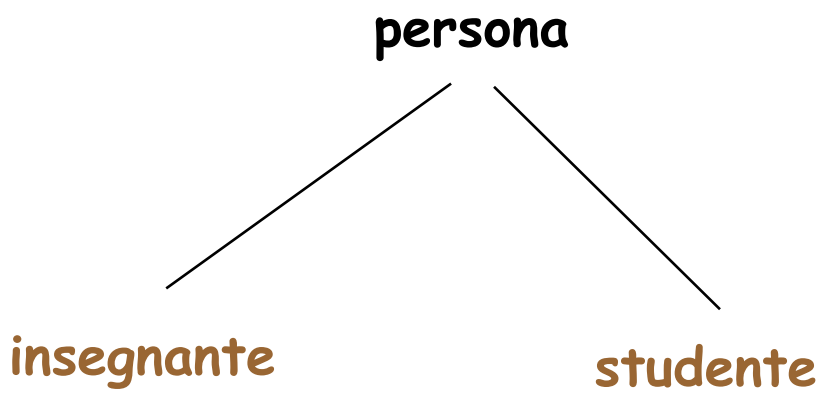
Ereditarietà di tipi

```
CREATE TYPE t_persona(
   nome CHAR(20),
   id       INTEGER,
   data_di_nascita    DATE,
   indirizzo      t_indirizzo);

CREATE TYPE t_insegnante(
   stipendio     DECIMAL(8,2),
   dipartimento        REF t_dipartimento,
   corsi_insegnati     TABLE OF REF t_corso)
UNDER t_persona;
```

Ereditarietà di tipi

```
CREATE TYPE t_studente(
   corsi_seguiti TABLE OF REF t_corso)
UNDER t_persona;
```



Ereditarietà di tabelle

- Come visto in precedenza i row type possono essere usati come tipi di tabelle
- I row type possono essere organizzati in gerarchie di ereditarietà
- La gerarchia definita sui row type impone anche una gerarchia sulle tabelle definite con questi tipi

Esempio

- CREATE TABLE Persone OF t_persona;
- CREATE TABLE Insegnanti of t_insegnante UNDER Persone;
- CREATE TABLE Studenti OF t_studente UNDER Persone;
- E’ stata creata una gerarchia tra le tabelle Persone, Insegnanti e Studenti
- La gerarchia influenza i risultati delle interrogazioni

Large Objects (LOBs)

- Facilitano la memorizzazione di dati multimediali (documenti, immagini, audio, ecc.)
- Possono contenere fino a 4GB di dati (di solito i RDBMS non vanno oltre 2-32KB)
- Il DBMS non associa nessuna interpretazione a questi dati

LOB

- Si distinguono in:
  - BLOB (Binary Large Object)
  - CLOB (Character Large Object)
- Sono fisicamente memorizzati esternamente alle tabelle ma internamente al DB (comportamento transazionale)

LOB

```
CREATE TABLE Pazienti(
    id                  INTEGER,
    età                 INTEGER,
    cartella_clinica    CLOB(1M),
    radiografia         BLOB(10M));
```

Metodi

- I metodi sono funzioni definite dall'utente associate ai tipi
- Possono essere scritti in linguaggi proprietari del DBMS o in 3GL
- La sintassi varia notevolmente a seconda del DBMS utilizzato

Metodi

```
CREATE TYPE t_persona(
   nome            CHAR(20),
   id                      INTEGER,
   data_di_nascita    DATE,
   indirizzo               t_indirizzo)
 MEMBER FUNCTION età(t_persona) RETURNS
   INTEGER;
```

Trigger

- I trigger sono delle regole attive (scritte in SQL) che possono essere associate alle tabelle
- I trigger vengono automaticamente attivati al verificarsi di una certa condizione
- La loro attivazione comporta una serie di operazioni nel DB (inserimenti in tabelle, ecc.)

Trigger

- Struttura generale:

  ON <evento>
  WHERE <condizione>
  DO <azione>

Trigger

- Evento: INSERT, DELETE, UPDATE
- Condizione: una qualsiasi condizione espressa in SQL (anche sottointerrogazioni)
- Azione: una qualsiasi istruzione SQL
- Tempo di attivazione: BEFORE o AFTER l'evento che ha causato l'attivazione del trigger

Esempio

- Si vuole definire un trigger che faccia in modo che ogni richiesta di rimborso per spese di viaggio contenga il codice del conto da cui tale rimborso deve essere effettuato
- Si supponga di avere due tabelle:
  - Spese(id_imp,id_dip,id_prg,inizio_viaggio,fine_viaggio,totale,n_conto)
  - Conti(n_conto,id_dip,id_prg)

Esempio

```
CREATE TRIGGER Codice_Conto
BEFORE INSERT on Spese
REFERENCING NEW AS N
FOR EACH ROW
WHEN(N.n_conto IS NULL)
   SET N.n_conto =
    (SELECT n_conto
    FROM Conti
    WHERE id_dip = N.id_dip
         AND id_prg = N.id_prg)
```

Trigger

- Consentono di specificare facilmente vincoli e controlli
- Un loro uso massiccio può però penalizzare le prestazioni
- Catene infinite di trigger

Stored Procedure

- Sono procedure memorizzate persistentemente nello schema del DB
- Possono essere invocate da molteplici applicazioni
- Migliorano le prestazioni in quanto riducono la comunicazione tra client (applicazione) e DBMS (server)

Stored Procedure

- Possono essere scritte in un linguaggio di programmazione (Java,C++,ecc.) o in un’estensione procedurale del linguaggio SQL (Oracle PL/SQL,Sybase Transact/SQL, Informix 4GL,)
- Il maggiore svantaggio delle stored procedure è la mancanza di standardizzazione