BASI DI DATI TEMPORALI

1. Motivazioni

2. Concetti fondamentali

3. Modelli dei dati temporali

4. Linguaggi di interrogazione temporali

5. TSQL2

6. OODAPLEX

MOTIVAZIONI

- Il *tempo* é un importante aspetto di molte applicazioni:
  - dati bancari
  - applicazioni mediche
  - sistemi di prenotazione di voli aerei
  - applicazioni finanziarie e commerciali
- basi di dati tradizionali inadeguate:
  - non rappresentano l'evoluzione dei dati nel tempo
  - vecchi valori sovrascritti dai nuovi

MOTIVAZIONI

BD convenzionali il tempo puó essere gestito a livello di *programma applicativo*

ESEMPIO:

Nelle BD *relazionali* : posso aggiungere due attributi indicanti l'istante di *inizio* e *fine* validitá delle informazioni nella tupla

SVANTAGGI:

la semantica di tali attributi é data dalle applicazioni $\Rightarrow$

- scrittura di molto codice complesso

- diminuzione efficienza

- aumento costi di verifica e manutenzione

Basi di dati temporali

SCOPO:

fornire un supporto alla memorizzazione e al reperimento di informazioni variabili nel tempo nel DBMS

VANTAGGI:

- semantica precisa e ben definita
- efficienza
- semplicitá
- espressivitá

Basi di dati temporali

IDEA:

Le modifiche sono viste come aggiunte alle informazioni presenti senza cancellazione delle informazioni preesistenti

Le basi di dati temporali sono viste come un'estensione delle basi di dati tradizionali:

- MODELLI DEI DATI: estensione dei modelli relazionale e orientato agli oggetti
- LINGUAGGIO DI QUERY (QL): estensioni dei linguaggi di query relazionali

Concetti fondamentali

- struttura del tempo

- densitá
  - modello discreto
  - modello denso
  - modello continuo

- dimensioni temporali

- tipi di dato temporali

Struttura del tempo

2 modelli strutturali:

Modello lineare: il tempo avanza dal passato al futuro in modo totalmente ordinato (*asse temporale*)

Modello ramificato: il tempo è lineare dal passato fino all'istante corrente, dove si divide in diversi assi temporali, ognuno rappresentante una potenziale sequenza di eventi (*albero*)

Struttura del tempo

Ulteriori classificazioni:

tempo _limitato_:

- sup. o inf. o da entrambe le parti
- limiti specificati esplicitamente o impliciti, di solito:

  limite inferiore ''Big-Bang''

  limite superiore ''Big-Crunch''

tempo _illimitato_

Densitá

Nel modello lineare si identificano le seguenti categorie:

- **Modello discreto**: isomorfo all'insieme dei numeri interi, ogni istante di tempo ha un unico successore

- **Modello denso**: isomorfo ai numeri razionali, presi due istanti qualsiasi di tempo, esiste sempre un numero finito di istanti compresi tra di essi

- **Modello continuo**: isomorfo ai numeri reali, presi due istanti qualsiasi di tempo, esiste sempre un numero infinito di istanti compresi tra di essi

Densitá

In generale si adotta il *modello discreto* in quanto è semplice e facilmente implementabile

Concetti importanti:

chronon : la più piccola entità temporale rappresentabile in un determinato modello (nel modello discreto i chronon sono i numeri interi)

granularità temporali : aggregazioni di chronon, per esempio i mesi, le settimane, i giorni ecc.

Dimensioni temporali

2 dimensioni temporali:

- **tempo di validitá**: tempo in cui il fatto è accaduto nella realtà, indipendentemente dal fatto che esso sia stato registrato o meno in qualche base di dati

- **tempo di transazione**: tempo durante il quale il fatto è presente nella base di dati, limiti inferiore e superiore identificati rispettivamente dalla transazione che lo ha inserito e da quella che lo ha rimosso

I tempi di validità e di transazione *sono ortogonali*, a volte possono anche coincidere.

Dimensioni temporali

Classificazione delle BD secondo la dimensione temporale supportata:

Snapshot: non supporta né il tempo di validità né quello di transazione, per esempio tutte le basi di dati convenzionali

Rollback: supporta il tempo di transazione, ma non il tempo di validità, è quindi possibile ripristinare lo stato che la base di dati aveva ad un determinato istante passato

Dimensioni temporali

Storica: supporta il tempo di validità ma non quello di transazione, contiene quindi la storia dei valori che i dati in essa memorizzati hanno assunto nel tempo

Bitemporale: supporta sia il tempo di validità che quello di transazione, è ad esempio possibile mantenere sia la storia dei valori che i dati hanno assunto nel tempo, sia mantenere traccia del tempo in cui le modifiche ai valori dei dati sono state effettivamente registrate nella base di dati

Dimensioni temporali

Altre dimensioni temporali (meno usate):

- *tempo di decisione*: denota il tempo in cui si è deciso che tale fatto doveva accadere

- *tempo definito dall'utente*: la semantica dei valori assunti da tale tipo di tempo è nota soltanto all'utente, e non è interpretata

Tipi di dato temporali

- *Singoli istanti*: è fondamentale poter disporre di tipi di dato che rappresentano singoli chronon, in quanto tali tipi di dato consentono di modellare il tempo di occorrenza di un determinato fatto

- *Periodi temporali*: denota l'insieme di istanti contenuti tra due istanti fissati. Un periodo può essere rappresentato tramite una coppia di attributi che assumono come valori singoli chronon; i valori di tali attributi rappresentano il limite superiore ed inferiore del periodo

  ESEMPIO:

  "Il primo Maggio del 1997"

Tipi di dato temporali

- *Intervalli temporali*: denota un lasso di tempo di durata fissa ma senza un preciso istante iniziale e finale. Un intervallo può essere rappresentato tramite un attributo che assume come valore un numero, rappresentante il numero di istanti che compongono l'intervallo.

  ESEMPIO:

  "Un giorno"

Tipi di dato temporali

- *Insiemi di istanti*: spesso può essere utile rappresentare insiemi finiti di istanti, non necessariamente contigui. Ad esempio, gli insiemi di istanti possono servire per rappresentare i tempi di occorrenza di un fatto che si ripete più di una volta.

- *Elementi temporali*: un elemento temporale è un insieme finito di periodi temporali, e quindi rappresenta un modo compatto per denotare un insieme di istanti non necessariamente contigui. Sono frequentemente utilizzati per rappresentare il tempo di validità e/o di transazione di un determinato dato.

Modelli dei dati temporali

Principalmente estensioni del modello relazionale o orientato agli oggetti.

Criteri di confronto:

1. le nozioni di tempo che tali modelli supportano (cioè tempo di validità, di transazione, ecc.)

2. come il tempo è associato alle relazioni o agli oggetti

3. come sono modellati i valori degli attributi temporali, cioè degli attributi che assumono valori variabili nel tempo

Modelli dei dati temporali

Come è associato il tempo alle relazioni o agli oggetti?

- Il tempo è associato all'intera tupla, o all'intero oggetto (modelli *omogenei*).
- Il tempo è associato a gruppi di attributi (modelli *eterogenei*).
- Il tempo è associato ai singoli attributi di una relazione o di un oggetto.
- Il tempo è associato a gruppi di tuple o di oggetti. Tale modalità è generalmente usata in modelli che supportano solo il tempo di transazione per identificare le tuple o gli oggetti che sono modificati o inseriti dalla stessa transazione.

Ogni approccio ha vantaggi e svantaggi ...

Modelli dei dati temporali

Come vengono rappresentati i valori degli attributi temporali?

- **Valori atomici**: non hanno alcuna struttura interna (es. `Verdi`) o sono insiemi di valori senza struttura interna (es. {`Verdi`,`Gialli`}).

- **Funzioni**: i valori di un attributo sono funzioni parziali dal dominio temporale all'insieme dei valori legali per l'attributo (es. `100` $\rightarrow$ `Verdi`, `101` $\rightarrow$ `Gialli`).

Modelli dei dati temporali

- **Coppie**: i valori sono coppie ordinate, o insiemi di coppie ordinate, in cui il primo elemento è un valore legale per l'attributo, o un insieme di valori legali per l'attributo, ed il secondo elemento è un istante di tempo
(es. `(Verdi,100)`, `({Verdi,Rossi},103)`).

- **Triple**: i valori sono triple, o insiemi di triple, il cui primo elemento è un valore legale per l'attributo, o un insieme di valori legali per l'attributo, ed i restanti elementi sono due istanti che determinano l'intervallo in cui l'attributo ha assunto quel determinato valore
(es. `(Gialli,101,109)`,
`({Rossi, Bianchi},105,120)`).

Linguaggi di interrogazione temporali

Proprietà che un linguaggio di query temporale dovrebbe possedere:

- appropriati costrutti per riferirsi al tempo di validità o di transazione di una tupla od oggetto all'interno di una query

- appropriati costrutti per poter identificare il limite inferiore e superiore del periodo temporale associato ad un determinato dato o generare un periodo o un intervallo, di una prefissata lunghezza, da due istanti di tempo specificati

- una serie di predicati predefiniti per confrontare istanti, periodi, intervalli ed elementi temporali

Linguaggi di interrogazione temporali

- meccanismi di *selezione* e *proiezione* temporale, cioè selezionare tuple od oggetti in base a specifiche condizioni sul loro tempo di validità o di transazione e che permettano di specificare il tempo di validità associato alle tuple od oggetti risultato di una interrogazione

- costrutti per esprimere facilmente *join temporali*, cioè che consentano di correlare dati rappresentati da relazioni od oggetti diversi, sulla base di condizioni sul loro tempo di validità o di transazione

- funzioni aggregate che coinvolgano la dimensione temporale

TSQL2

- *Temporal Structured Query Language*, estensione temporale del linguaggio di interrogazione per basi di dati tradizionali SQL2

- obiettivi:

  - piena compatibilità con il linguaggio SQL2

  - superare i limiti di tale linguaggio per quanto riguarda la gestione di informazioni variabili nel tempo

  - diventare il linguaggio standard di interrogazione per una base di dati temporale

Modello concettuale bitemporale

- basato su un modello del tempo discreto, isomorfo all'insieme dei numeri naturali

- tempo associato all'intera tupla

- supporta sia il tempo di validità sia quello di transazione $\Rightarrow$ due domini temporali: quello del tempo di validità $D_v = \{t_1, \ldots, t_k\}$, dove $t_1, \ldots t_k \in \mathbb{N}$, e quello del tempo di transazione $D_t = \{t'_1, \ldots, t'_j\} \cup \{UC\}$

- alle tuple sono associati elementi bitemporali, che sono insiemi di chronon bitemporali

  ogni chronon bitemporale rappresenta un rettangolino nello spazio bitemporale tempo di validità/tempo di transazione

Tipi di dato temporali

- TSQL2 supporta tutti i tipi di dato temporali forniti da SQL2 (`date`, `time` e `timestamp`, per singoli istanti, `interval` per intervalli temporali) + tipo `period` per periodi temporali

- all'interno delle interrogazioni si possono riferire sia elementi temporali sia insiemi di istanti

- *operatori temporali* per manipolare e confrontare attributi con tipo temporale - tre categorie: costruttori, estrattori e operatori di paragone

Operatori temporali

- *costruttori* - esempi: `PERIOD` (dati due istanti costruisce un periodo), `INTERSECT` (intersezione di periodi), `INTERVAL` (dato un periodo ne restituisce la durata)

- *estrattori* - esempi: `BEGIN` e `END` (istante iniziale e finale di un periodo), `FIRST` e `LAST` (primo e ultimo periodo di un elemento temporale)

- *operatori di paragone*: `PRECEDES`, =, `OVERLAPS`, `CONTAINS` e `MEETS` (per il confronto di elementi temporali, periodi ed istanti) e =, $<$, $>$ (per il confronto di intervalli)

Creazione di relazioni

comando `CREATE TABLE` di SQL2 esteso per specificare il tipo temporale della relazione - clausola `AS`

`[[AS VALID [STATE | EVENT]` *Granularità* `[AND TRANSACTION]] |`

`[AS TRANSACTION]]);`

- se la clausola `AS` non è specificata, la relazione è una relazione snapshot (non è mantenuta nessuna informazione temporale)

- se si specifica `AS VALID` o `AS VALID STATE` (notazioni equivalenti), la relazione mantiene il tempo di validità (rappresentato mediante un elemento temporale)

- se si specifica `AS VALID EVENT`, la relazione mantiene il tempo di validità rappresentato mediante insiemi di istanti

  sia in questo caso che nel precedente è possibile specificare la granularità temporale che si vuole utilizzare

- se alle clausole precedenti viene aggiunta la parola chiave `TRANSACTION`, la relazione mantiene anche il tempo di transazione

  i tempi di validità e di transazione sono memorizzati mediante insiemi di coppie il (tempo di transazione, tempo di validità)

  la granularità del tempo di transazione è definita dal sistema e non può essere specificata esplicitamente

- se si specifica solo `AS TRANSACTION`, la relazione mantiene solo il tempo di transazione (rappresentato mediante un elemento temporale)

Esempio

```
CREATE TABLE Impiegati
       (Imp#            Decimal,
        Nome            Char,
        Stipendio       Decimal,
        Dip#            Decimal
AS VALID DAY);
```

Clausola FROM

pb. piu' tuple nel risultato legate a variazioni su attributi su cui non si proietta (es. `Salario`)

```
SELECT Nome
FROM Impiegati(Nome,Dip#)
WHERE Dip# = 1001;
```

ha l'effetto di costruire una relazione, con colonne `Nome` e `Dip#` ottenuta selezionando tutte le tuple della relazione `Impiegati` aventi gli stessi valori per gli attributi `Nome` e `Dip#` e collassando gli elementi temporali ad esse associati in un unico elemento temporale

Selezione temporale

- selezione di tuple sia in base al loro tempo di validità che al loro tempo di transazione

- operatori `VALID()` e `TRANSACTION()` nella clausola `WHERE` di una interrogazione

esempio: ritrovare i nomi di tutti gli impiegati che hanno ricevuto un aumento di stipendio il primo Marzo 1997

```
SELECT I1.Nome
FROM Impiegati AS I1, Impiegati AS I2
WHERE I1.Imp# = I2.Imp# AND
      I2.Stipendio > I1.Stipendio AND
      VALID(I1) MEETS VALID(I2) AND
      END(VALID(I1)) = Date `01/03/97´;
```

esempio: selezionare gli identificatori di tutti gli impiegati il cui cambiamento di stipendio è stato memorizzato nella base di dati il 1 Luglio 1996

SELECT I_1.Imp#
FROM Impiegati AS I_1, Impiegati AS I_2
WHERE I_1.Imp# $=$ I_2.Imp# AND
      I_1.Stipendio $\neq$ I_2.Stipendio AND
      TRANSACTION(I_1) MEETS TRANSACTION(I_2) AND
      CAST(BEGIN(TRANSACTION(I_2)) AS DAY) $=$
      Date `01/07/96´;

Proiezione temporale

- due clausole: `VALID` e `VALID INTERSECT`, che devono essere seguite da una espressione temporale, che può denotare sia un periodo sia un elemento temporale

- servono per specificare il tempo di validità delle tuple risultato dell'interrogazione, calcolato in base all'espressione temporale specificata

- clausola `VALID`: permette di assegnare un tempo di validità arbitrario alle tuple risultato dell'interrogazione

- clausola `VALID INTERSECT`: il tempo di validità delle tuple risultato dell'interrogazione è dato dall'intersezione dell'espressione temporale specificata con il tempo di validità associato alle tuple estratte

Esempio

si vuole creare un nuovo dipartimento 1001001 tale che tutti gli impiegati che attualmente lavorano nel dipartimento 1001 ed il cui stipendio è maggiore di due milioni siano trasferiti nel nuovo dipartimento a partire da un mese dopo l'istante corrente:

```
INSERT INTO Dipartimenti(Imp#,Dip#)
SELECT Imp#, 1001001
VALID PERIOD(CURRENT_DATE + INTERVAL
      `1´ MONTH, Date `Forever´)
FROM Impiegati
WHERE Dip# = 1001 AND Stipendio > 2000 AND
      Impiegati OVERLAPS CURRENT_DATE;
```

Esempio

determinare i dipartimenti nei quali l'impiegato `Rossi` ha lavorato durante il 1994

```
SELECT Dip#
VALID INTERSECT(Impiegati, PERIOD `[1994]´
DAY)
FROM Impiegati WHERE Nome = `Rossi´;
```

risultato è lista di identificatori di dipartimenti, ognuno con associato l'insieme di periodi durante il 1994 in cui `Rossi` vi ha lavorato

Funzioni aggregate

- le funzioni aggregate di SQL2 (`MIN`, `MAX`, `COUNT`, `SUM` e `AVG`) sono estese in TSQL2 in modo che possano essere applicate a domini temporali e per restituire relazioni temporali

  esempio:

  ```
  SELECT COUNT(*)
  FROM Impiegati
  WHERE Dip# = 1001;
  ```

  variazione del numero di impiegati del dipartimento 1001

- aggregato temporale, `RISING` che calcola il massimo periodo in cui il valore di un attributo è cresciuto in modo monotono

  esempio:

  ```
  SELECT RISING(Stipendio)
  FROM Impiegati
  WHERE Imp# = 104;
  ```

  il più lungo periodo in cui lo stipendio dell'impiegato `104` è cresciuto monotonicamente.

- se la parola chiave `WEIGHTED` compare subito dopo il nome della funzione aggregata, i valori passati in ingresso alla funzione sono *pesati* in base al loro periodo di validità

- la clausola `GROUP BY` è stata estesa per permettere il *raggruppamento temporale*: consente di definire una partizione della linea temporale, raggruppare le tuple della relazione secondo tale partizione e computare la funzione aggregata sui gruppi risultanti

  clausola `USING` che indica la partizione della linea temporale che si vuole adottare per il raggruppamento

  <u>esempio:</u>

  per ogni impiegato, si determini la media trimestrale dello stipendio

```
SELECT AVG(Stipendio)
FROM Impiegati AS I
GROUP BY  VALID(I) USING 3 MONTH;
```

OODAPLEX

Modello dei dati

```
type Impiegato is Persona
function Imp# (i:Impiegato → id:integer)
function Nome (i:Impiegato → n:string)
function Stipendio (i:Impiegato →
    f:(t:time → i:integer))
function Dip (i:Impiegato →
    f:(t:time → dip:Dipartimento))

type Dipartimento is object
function Dip# (d:Dipartimento → id:integer)
function Nome (d:Dipartimento → n:string)
function Ufficio (d:Dipartimento → u:string)
function Divisione (d:Dipartimento → d:string)
function Dirigente (d:Dipartimento →
    f:(t:time → dir:Impiegato))
```

tempo associabile sia agli oggetti che agli attributi

```
type Impiegato is object
function Stato (i:Impiegato →
    f:(t:time → s:Imp_Statico))

type Imp_Statico is object
function Imp# (s:Imp_Statico → id:integer)
function Nome (s:Imp_Statico → n:string)
function Stipendio (s:Imp_Statico → i:integer)
function Dip (s:Imp_Statico → dip:Dipartimento)
```

OODAPLEX

Linguaggio di interrogazione

- lo stipendio dell'impiegato `Rossi` quando lavorava per il dipartimento 1001:

```
FOR THE i IN extent(Impiegato)
    WHERE Nome(i) = `Rossi´
    FOR EACH t WHERE Dip#(Dip(i)(t)) = 1001
        Stipendio(i)(t)
    END
END
```

- istanti in cui `Rossi` ha avuto un aumento di stipendio, e il valore di tale aumento:

```
FOR THE i IN extent(Impiegato)
    WHERE Nome(i) = `Rossi´
    FOR EACH t IN lifespan(i)
    WHERE Stipendio(i)(t) > Stipendio(i)(t-1)
        [tempo:t,aumento:
        Stipendio(i)(t) - Stipendio(i)(t-1)]
    END
END
```

- somma totale finora percepita da `Rossi` come stipendio:

```
FOR THE i IN extent(Impiegato)
    WHERE Nome(i) = `Rossi´
    FOR EACH t IN lifespan(i)
        SUM(⟨Stipendio(i)(t)/365⟩)
    END
END
```

- per ciascun impiegato, determinare il primo giorno dopo cui lo stipendio non è più diminuito:

$$
\begin{array}{l}
\texttt{FOR EACH } i \texttt{ IN extent(Impiegato)} \\
\quad \texttt{MIN(}\langle t \texttt{ IN lifespan(}i\texttt{) WHERE} \\
\qquad \texttt{FOR ALL } t' > t \\
\qquad\quad \texttt{(Stipendio(}i\texttt{)(}t'\texttt{+1)} \geq \\
\qquad\qquad \texttt{Stipendio(}i\texttt{)(}t'\texttt{))}\rangle\texttt{)} \\
\texttt{END}
\end{array}
$$