# Vincoli di Integrità

**Massimo Mecella**

Dipartimento di Ingegneria informatica automatica e gestionale
Antonio Ruberti
Sapienza Università di Roma

**Progetto di Applicazioni Software**

# Generalità

- I vincoli di integrità specificano proprietà dei dati
- I vincoli **restringono** l'insieme delle istanze possibili su uno schema, allo scopo di riflettere meglio il dominio che i dati medesimi rappresentano
- Nella definizione di uno schema, occorre specificare tutti i vincoli che devono sussistere sulle istanze dello schema
- Occorre specificare vincoli di integrità:
  - sullo schema concettuale
  - sullo schema logico

# Vincoli sullo schema concettuale

- Vincoli di identificazione
- Vincoli di cardinalità
- Vincoli esterni

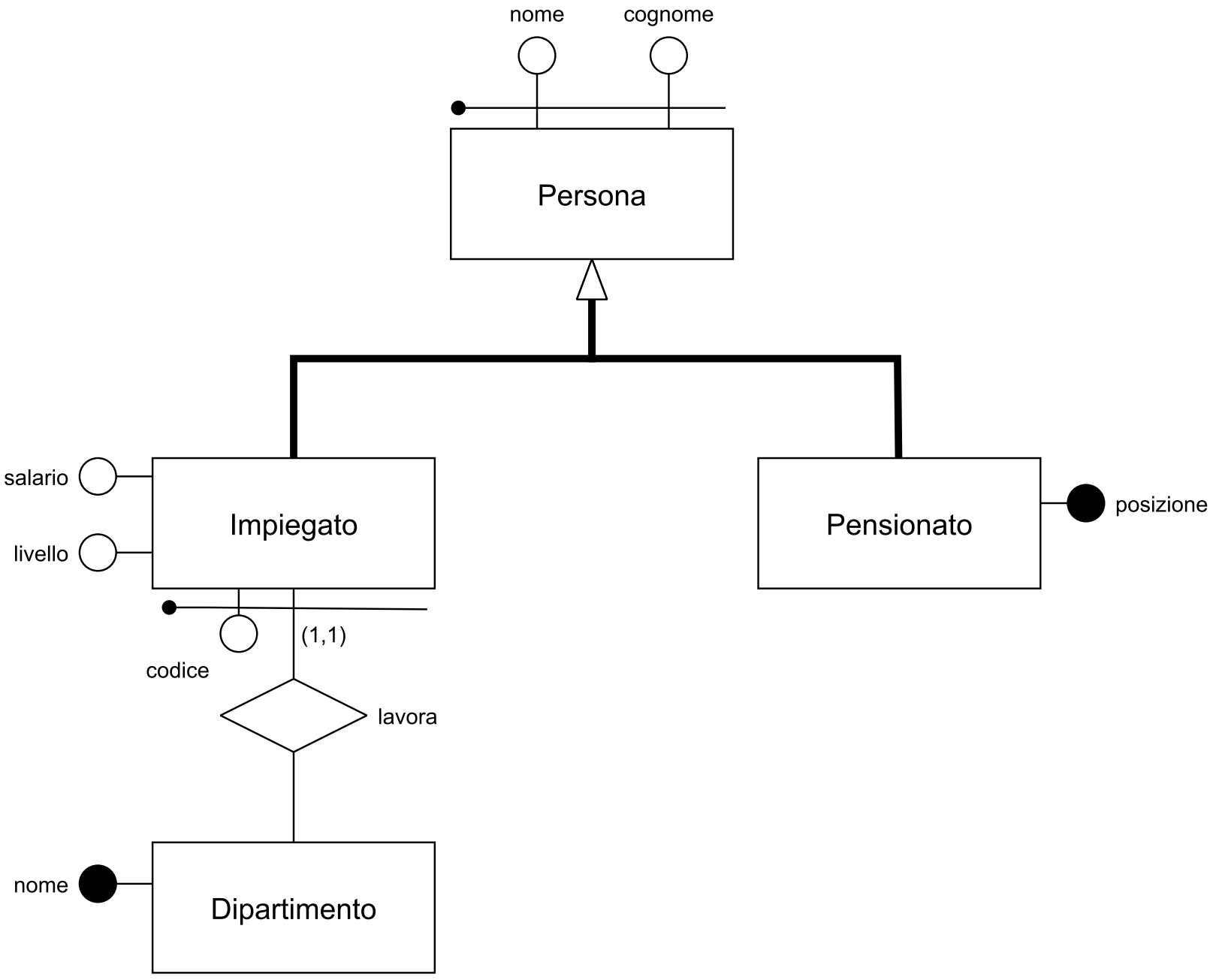
# Prima classificazione dei vincoli di integrità

Sullo schema concettuale, possiamo classificare i vincoli a seconda che esprimano proprietà che devono essere soddisfatte da ogni stato, coppia di stati consecutivi, o insieme di stati dei dati:

- vincoli **di stato**: si riferiscono ad ogni stato dei dati
- vincoli **di transizione di stato**: si riferiscono a due stati consecutivi dei dati
- vincoli **di sequenza di stati (temporali)**: si riferiscono a più di due stati (non necessariamente consecutivi) dei dati

# Vincoli di stato

Specificano delle proprietà statiche dei dati. Sia i vincoli di identificazione che i vincoli di cardinalità sono vincoli di stato. Si consideri il seguente schema ER.


nome
cognome
Persona
salario
livello
Impiegato
Pensionato
posizione
codice
(1,1)
lavora
nome
Dipartimento

Esempi di vincoli esterni che possono essere classificati come vincoli di stato sono i seguenti.

- Il livello di un impiegato deve essere un intero tra 1 e 8.
- La somma dei salari di tutti gli impiegati non può superare un tetto $M$.

# Vincoli di transizione di stato

Questi vincoli riguardono due stati consecutivi dei dati. Facendo riferimento alla precedente entità `Impiegato`, esempi di vincoli esterni che possono essere classificati di transizione di stato sono i seguenti:

- Il salario di un impiegato non può essere incrementato di più del 5 per cento con una singola operazione di aggiornamento.
- Il salario medio non può essere incrementato di più del 5 per cento con una singola operazione di aggiornamento.
- Un impiegato può diventare un pensionato ma non viceversa.

# Vincoli temporali

Questi vincoli riguardono un insieme di più di due stati non necessariamente consecutivi. Ad esempio (facendo sempre riferimento allo scenario precedente), vincoli esterni che possono essere classificati come vincoli temporali:

- Una volta che un impiegato è stato licenziato da un dipartimento, non può più essere impiegato in quel dipartimento.
- Il salario medio di un impiegato non può incrementare di oltre il 10% in meno di 10 operazioni di aggiornamento.
- Prima di diventare pensionato, un impiegato deve aver assunto almeno 3 diversi livelli di impiego.

# Vincoli sullo schema logico

Nel passaggio dallo schema concettuale allo schema logico, i vincoli sono anch'essi tradotti in vincoli espressi in termini dello schema logico (i.e., relazionale).
Lo schema relazionale risultante dalla traduzione è quindi corredato da un insieme di vincoli:

- vincoli di chiave (primaria e no) - risultanti, per esempio, dalla traduzione dei vincoli di identificazione o da cardinalità massime pari a 1 (i.e., funzionalità);
- attributi obbligatori - risultanti dalla traduzione dei vincoli di cardinalità minima pari a 1;
- vincoli di chiave esterna (foreign key) - risultanti, per esempio, da tipizzazione di relazioni, da vincoli esterni introdotti sull'ER dopo ristrutturazione delle ISA, o da decomposizioni di tabelle nella fase di ristrutturazione del logico;

- vincoli di inclusione - risultanti, per esempio, dalla traduzione di vincoli di cardinalità minima pari a 1 sulle relazioni;
- vincoli di disgiuntezza - risultanti, per esempio, dalla traduzione di generalizzazioni;
- vincoli di completezza - risultanti, per esempio, dalla traduzione di generalizzazioni complete;
- altri vincoli esterni - risultanti dalla traduzione di vincoli esterni espressi sull'ER.

# Ulteriore classificazione dei vincoli di integrità

Sullo schema logico, è possibile classificare i vincoli, oltre che secondo il criterio precedente, a seconda che esprimano proprietà che devono essere soddisfatte da ogni tupla, tabella o insieme di tabelle:

- vincoli **di tupla (riga)**: si riferiscono alle singole righe di una tabella
- vincoli **di tabella**: si riferiscono all'insieme delle righe di una tabella
- vincoli **inter-tabella**: sono i più generali, e si riferiscono a più di una tabella

# Vincoli di tupla

Specificano delle proprietà che devono essere soddisfatte da ogni singola tupla di una tabella.
I vincoli di attributi obbligatori sono vincoli di tupla.
Siano ora `Impiegato`, `Dipartimento` e `Pensionato` le tabelle dello schema logico derivanti dalla traduzione della porzione dello schema ER vista in precedenza (sono state accorpate nella tabella `Impiegato` sia la ER-relazione `lavora` che la ISA derivante dalla generalizzazione). La tabella `Impiegato` ha attributi `Codice`, `Nome`, `Cognome`, `Livello`, `Salario`, `Dipartimento`. I seguenti vincoli, derivanti anch'essi dalla traduzione dei vincoli sull'ER, sono ulteriori esempi di vincoli di tupla:

- Il livello di un impiegato deve essere un intero tra 1 e 8.
- Il salario di un impiegato non può essere incrementato di più del 5 per cento con una singola operazione di aggiornamento.

- Una volta che un impiegato è stato licenziato da un dipartimento, non può più essere impiegato in quel dipartimento.

# Vincoli di tabella

Specificano delle proprietà che devono essere soddisfatte dall'insieme delle tuple di una tabella.
I vincoli di chiave sono vincoli di tabella.
Facendo riferimento alla tabella `Impiegato`, esempi di vincoli di tabella sono:

- La somma dei salari di tutti gli impiegati non può superare un tetto $M$.
- Il salario medio non può essere incrementato di più del 5 per cento con una singola operazione di aggiornamento
- Il salario medio di un impiegato non può incrementare di oltre il 10% in meno di 10 operazioni di aggiornamento.

# Vincoli inter-tabella

Specificano delle proprietà che devono essere soddisfatte dalle tuple di un insieme di tabelle.
I vincoli di chiave esterna e di inclusione sono vincoli inter-tabella.

- Un impiegato può diventare un pensionato ma non viceversa.
- Prima di diventare pensionato, un impiegato deve aver assunto almeno 3 diversi livelli di impiego.

# Specifica dei vincoli

- È opportuno che i vincoli siano specificati in modo **dichiarativo**, e cioè facendo riferimento a **proprietà** dei dati, e non a operazioni che consentano il soddisfacimento di dette proprietà. Si può usare una descrizione precisa in linguaggio naturale, oppure formalismi ad-hoc
- I vincoli devono essere **decomposti** in vincoli elementari, in modo che possano essere facilmente implementati (come vedremo più avanti)

## Consistenza dei vincoli

- Un insieme di vincoli si dice **consistente** se esiste almeno un'istanza dello schema che li verifica tutti.
- In molti casi non è difficile definire vincoli non consistenti.

# Ridondanza

- Un insieme di vincoli si dice **ridondante** se esiste in esso almeno un vincolo che implica un altro.
- È opportuno che non ci sia ridondanza nei vincoli definiti su uno schema.

# Implementazione dei vincoli

- **Implementare** i vincoli significa assicurare che i dati si trovino sempre in uno stato consistente rispetto ai vincoli.
- In linea di principio, i vincoli devono essere implementati al **livello di gestione dei dati** (tramite gli strumenti del DBMS), affinché siano soddisfatti indipendentemente dalla modalità con cui si fa loro accesso.
- Quando, per scelta o a causa di limiti del DBMS, alcuni vincoli non sono implementati sulla base di dati, è necessario documentare in maniera adeguata la base di dati e delegare l'implementazione dei suddetti vincoli alle applicazioni che accedono alla stessa.

# Vincoli di integrità nel linguaggio SQL

Per **dichiarare** vincoli di integrità possiamo usare i seguenti costrutti SQL:

- Costrutto `NOT NULL` : permette di definire attributi obbligatori
- Costrutto `UNIQUE`: permette di definire uno o più attributi che identificano la tupla
- Costrutto `PRIMARY KEY`: permette di definire uno o più attributi che formano una chiave primaria (implica `NOT NULL`)
- Costrutto `FOREIGN KEY` (`REFERENCES`): permette di definire chiavi esterne
- Costrutto `CHECK`: permette di definire vincoli generici intra-tabella
- Costrutto `CREATE ASSERTION`: permette di definire vincoli generici inter-tabella

Tranne `CREATE ASSERTION`, gli altri costrutti sono usati all'interno dei comandi `CREATE TABLE` e `ALTER TABLE`.

# I costrutti NOT NULL, UNIQUE e PRIMARY KEY (esempio)

```
CREATE TABLE IMPIEGATO (
CODICE INT,
NOME VARCHAR(20) NOT NULL,
COGNOME VARCHAR(20) NOT NULL,
LIVELLO INT NOT NULL,
SALARIO INT NOT NULL,
DIPARTIMENTO INT NOT NULL,
PRIMARY KEY (CODICE, DIPARTIMENTO),
UNIQUE (NOME, COGNOME)
);
```

# Il costrutto FOREIGN KEY

- Consente di esprimere un vincolo di integrità referenziale (ad es., $R_1[A] \subseteq R_2[B]$, con $B$ **chiave** di $R_2$)
- $B$ può essere chiave primaria (`PRIMARY KEY`) o secondaria (`UNIQUE`)
- a partire da SQL99 consente di definire una **politica di reazione alle modifiche** (cancellazioni ed aggiornamenti) dei dati contenuti nella tabella esterna ($R_2$) che possono portare alla violazione del vincolo

# Il costrutto FOREIGN KEY (cont.)

- per le cancellazioni si fa uso della clausola

```
ON DELETE [CASCADE|SET NULL|SET DEFAULT|
                                NO ACTION|RESTRICT]
```

  a fronte della cancellazione di una tupla nella tabella esterna, tutte le righe della tabella interna che referenziano la tupla cancellata vengono eliminate (opzione `CASCADE`), poste a `NULL` (`SET NULL`), poste al valore di default (`SET DEFAULT`). Con l'opzione `RESTRICT`, se il vincolo risulta violato, si impedisce la cancellazione.

# L'opzione NO ACTION

Le opzioni NO ACTION e RESTRICT sono tra loro molto simili. Infatti, se la verifica del vincolo fallisce, sia in caso di utilizzo dell'opzione NO ACTION che in caso di utilizzo di RESTRICT, l'operazione porta ad un errore.
La principale differenza tra le due è che con l'opzione NO ACTION, la verifica del vincolo di integrità è fatta dopo aver tentato di modificare la tabella esterna, mentre con l'opzione RESTRICT la verifica è fatta prima di tentare l'esecuzione del DELETE.

# Il costrutto FOREIGN KEY (cont.)

- per gli aggiornamenti si fa uso della clausola

```
ON UPDATE [CASCADE|SET NULL|SET DEFAULT|
                                NO ACTION|RESTRICT]
```

  a fronte dell'aggiornamento della chiave della tabella esterna, all'attributo delle corrispondenti tuple della tabella interna viene assegnato lo stesso nuovo valore (`CASCADE`), il valore `NULL` (`SET NULL`) oppure il valore di default (`SET DEFAULT`). Con l'opzione `RESTRICT` si impedisce l'aggiornamento. L'opzione `NO ACTION` ha la stessa semantica descritta per le cancellazioni.

# Il costrutto FOREIGN KEY (esempio)

```
CREATE TABLE IMPIEGATO (
CODICE INT,
NOME VARCHAR(20) NOT NULL,
COGNOME VARCHAR(20) NOT NULL,
LIVELLO INT NOT NULL ,
SALARIO INT NOT NULL,
DIPARTIMENTO INT NOT NULL
                    REFERENCES DIPARTIMENTO (COD_DIP)
                    ON DELETE SET NULL
                    ON UPDATE CASCADE,
PRIMARY KEY (CODICE, DIPARTIMENTO),
UNIQUE (NOME, COGNOME)
);
```

# Il costrutto DEFAULT

- Nei casi in cui per specificare la reazione ad una cancellazione o ad una modifica si ricorre alla clausola `SET DEFAULT`, è necessario che sia previsto un valore di default da assegnare all'attributo che eventualmente rimane "appeso".
  $\rightarrow$ Nell'istruzione `CREATE TABLE` si usa la seguente sintassi:

  `<Nome Colonna> <tipo SQL> DEFAULT <expression>`

- Quando si definisce un valore di default per una colonna, tale valore viene assegnato quando si inserisce una tupla senza specificare alcun valore per l'attributo in oggetto

# Il costrutto DEFAULT (esempio)

```
CREATE TABLE IMPIEGATO (
CODICE INT,
NOME VARCHAR(20) NOT NULL,
COGNOME VARCHAR(20) NOT NULL,
LIVELLO INT DEFAULT 1 NOT NULL ,
SALARIO INT NOT NULL,
DIPARTIMENTO INT NOT NULL
                    REFERENCES DIPARTIMENTO (COD_DIP)
                    ON DELETE SET NULL
                    ON UPDATE CASCADE,
PRIMARY KEY (CODICE, DIPARTIMENTO),
UNIQUE (NOME, COGNOME)
);
```

# Il costrutto CHECK

- Consente di esprimere vincoli di integrità complessi su una sola tabella:

  `CHECK (<condizione>)`

  dove `<condizione>` è una qualsiasi condizione che può comparire all'interno di una clausola `WHERE` (incluse quelle che fanno uso di sub-query)
- Consente di implementare vincoli a livello di tupla (ad esempio, `SALARIO >= 0`) e di tabella (in genere tramite sub-query con operatori aggregati).

# Esempio

```
CREATE TABLE IMPIEGATO (
CODICE INT,
NOME VARCHAR(20) NOT NULL,
COGNOME VARCHAR(20) NOT NULL,
LIVELLO INT DEFAULT 1 NOT NULL
                    CHECK (LIVELLO <= 8 AND LIVELLO >=1),
SALARIO INT NOT NULL CHECK (SALARIO >=0),
DIPARTIMENTO INT NOT NULL
                    REFERENCES DIPARTIMENTO (COD_DIP),
PRIMARY KEY (CODICE, DIPARTIMENTO),
UNIQUE (NOME, COGNOME),
CHECK (50 > (SELECT AVG(SALARIO) FROM IMPIEGATO))
);
```

# Il comando `ALTER TABLE`

In alternativa alla sintassi descritta fin qui per definire vincoli, si può usare il comando `ALTER TABLE`:

```
ALTER TABLE <Nome tabella> ADD <{PRIMARY KEY |
                        FOREIGN KEY | UNIQUE | CHECK}>
```

Tale comando è utile per realizzare vincoli di integrità referenziali ciclici: per far sì che $R_1$ referenzi $R_2$ ed $R_2$ referenzi $R_1$ si può definire prima $R_1$ senza vincolo di foreign key (altrimenti si dovrebbe far riferimento ad R2 che non è stata ancora definita), poi $R_2$ con il vincolo di foreign key verso $R_1$, ed infine aggiungere il vincolo di foreign key ad $R_1$ con il comando `ALTER TABLE`

# Il costrutto ASSERTION

- Per i vincoli inter-tabella, il costrutto `CREATE ASSERTION` permette di definire vincoli a livello di schema:

```
CREATE ASSERTION <NomeAss> CHECK (<condizione>);
```

# Esempio (vincolo di inclusione)

**vincolo:** Ogni impiegato risiede in almeno una città (supponiamo che esista una tabella `RISIEDE_IN(IMPIEGATO,CITTA)`, con chiave su entrambe le colonne)

```
CREATE ASSERTION impiegato_risiede
  CHECK (NOT EXISTS (SELECT * FROM IMPIEGATO
                     WHERE CODICE NOT IN
                     (SELECT IMPIEGATO FROM RISIEDE_IN)
  );
```

# Assegnare nomi espliciti a vincoli in SQL

Tutti i vincoli di integrità (tranne i vincoli sugli attributi obbligatori) hanno un **nome unico all'interno di uno schema**.
Se il nome non è specificato esplicitamente, viene assegnato dal sistema secondo una modalità dipendente dall'implementazione.
Esplicitare il nome di un vincolo può essere conveniente, per esempio, quando si prevede la necessità di eliminare il vincolo stesso (con l'istruzione `DROP CONSTRAINT <Nome Vincolo>`)
Per esplicitare il nome di un vincolo:

- nell'istruzione `CREATE TABLE`, si usa la sintassi

  ```
  CONSTRAINT <NomeVincolo> [ PRIMARY KEY | UNIQUE |
  FOREIGN KEY | CHECK ];
  ```

- nell'istruzione `ALTER TABLE`, si usa la sintassi

  ```
  ADD CONSTRAINT <NomeVincolo> [ {PRIMARY KEY | UNIQUE |
  FOREIGN KEY | CHECK} ];
  ```

```
CREATE TABLE IMPIEGATO (
NOME VARCHAR(20) NOT NULL,
COGNOME VARCHAR(20) NOT NULL,
LIVELLO INT CHECK (LIVELLO <= 8 AND LIVELLO >=1),
SALARIO INT CHECK (SALARIO >=0),
DIPARTIMENTO INT,
CONSTRAINT imp_fk FOREIGN KEY (DIPARTIMENTO)
      REFERENCES DIPARTIMENTO(COD_DIP),
CONSTRAINT imp_pk PRIMARY KEY (CODICE, DIPARTIMENTO),
CONSTRAINT imp_k UNIQUE (NOME, COGNOME),
CONSTRAINT imp_max
      CHECK (50 > (SELECT AVG(SALARIO) FROM IMPIEGATO))
);
```

# Modalità di verifica di vincoli

Ogni vincolo è caratterizzato da una **modalità di verifica** (checking mode), che determina il momento in cui il vincolo viene verificato: nel caso in cui la modalità sia `IMMEDIATE` il vincolo è controllato dopo l'esecuzione di ogni comando SQL; se la modalità è `DEFERRED` il vincolo è verificato solo al termine della transazione corrente

La modalità deferred può essere applicata solo ai vincoli che, al momento della loro creazione, sono stati definiti `DEFERRABLE` (`N.B.` Se non si specifica nulla, un vincolo viene implicitamente definito come `NON DEFERRABLE`).

- Se un vincolo è inizialmente definito `DEFERRABLE`, allora si deve specificare la sua modalità di verifica iniziale:

```
CONSTRAINT <NomeVincolo> [ PRIMARY KEY | UNIQUE |
FOREIGN KEY | CHECK ]
[DEFERRABLE
INITIALLY IMMEDIATE | INITIALLY
DEFERRED | NON DEFERRABLE];
```

- per modificare la modalità di verifica di un vincolo inizialmente definito DEFERRABLE, si usa la sintassi:

```
SET CONSTRAINT <NomeVincolo> [DEFERRED | IMMEDIATE];
```

# Esempio

```
CREATE TABLE IMPIEGATO (
CODICE INT,
NOME VARCHAR(20),
COGNOME VARCHAR(20),
LIVELLO INT CHECK (LIVELLO <= 8 AND LIVELLO >=1),
SALARIO INT CHECK (SALARIO >=0),
DIPARTIMENTO INT,
PRIMARY KEY (CODICE, DIPARTIMENTO),
UNIQUE (NOME, COGNOME),
CHECK (50 > (SELECT AVG(SALARIO) FROM IMPIEGATO))
CONSTRAINT imp_fk FOREIGN KEY (DIPARTIMENTO)
                  REFERENCES DIPARTIMENTO(COD_DIP)
                  DEFERRABLE INITIALLY IMMEDIATE
);
SET CONSTRAINT imp_fk DEFERRED;
```

# Un'alternativa al costrutto CHECK

Nei casi in cui si voglia forzare il valore di alcuni campi ad appartenere da un determinato insieme finito, è possibile usare un tipo enumerato ENUM.

```
CREATE TABLE PERSONA(
NOME    VARCHAR(20),
COGNOME VARCHAR(20),
ETA     INT,
SESSO   ENUM ('M','F')
);
```

In questo esempio, posso inserire tuple che abbiano nel campo SESSO esclusivamente i valori 'M','F','m','f'.