

# Vincoli, procedure e regole attive in SQL

# Qualità dei dati

- **Qualità dei dati:**  
correttezza, completezza, attualità
- In molte applicazioni reali i dati sono di scarsa qualità (5% - 40% di dati scorretti)
- Per aumentare la qualità dei dati:
  - Regole di integrità
  - Manipolazione dei dati tramite programmi predefiniti (procedure e trigger)

# Vincoli di integrità generici

- **Predicati che devono essere veri se valutati su istanze corrette (legali) della base di dati**
- **Espressi in due modi:**
  - **negli schemi delle tabelle**
  - **come asserzioni separate**

**Negli schemi delle tabelle si utilizza la clausola:**

**CHECK (PREDICATO)**

**associata ai vari attributi oppure espressa al termine della dichiarazione della tabella**

# Esempio : gestione magazzino

## magazzino

<b>COD-PROD</b>	<b>QTA-DISP</b>	<b>QTA-RIORD</b>
<b>1</b>	<b>150</b>	<b>100</b>
<b>3</b>	<b>130</b>	<b>80</b>
<b>4</b>	<b>170</b>	<b>50</b>
<b>5</b>	<b>500</b>	<b>150</b>

## riordino

<b>COD-PROD</b>	<b>DATA</b>	<b>QTA-ORD</b>

# Esempio: definizione di MAGAZZINO

```
CREATE TABLE MAGAZZINO AS
( COD-PROD CHAR(2) PRIMARY KEY,
  QTA-DISP INTEGER NOT NULL
    CHECK(QTA-DISP>10),
  QTA-RIORD INTEGER NOT NULL
    CHECK (QTA-RIORD>10),
    CHECK (QTA-DISP>QTA-RIORD) )
```

# Asserzioni

- **Predicati espressi separatamente dalla definizione delle tabelle, che devono essere veri se valutati su istanze corrette (legali)**

```
CREATE ASSERTION Ordini-Limitati AS  
CHECK( 1000 >=  
        SELECT COUNT(COD-PROD)  
        FROM RIORDINO )
```

# Significato dei vincoli

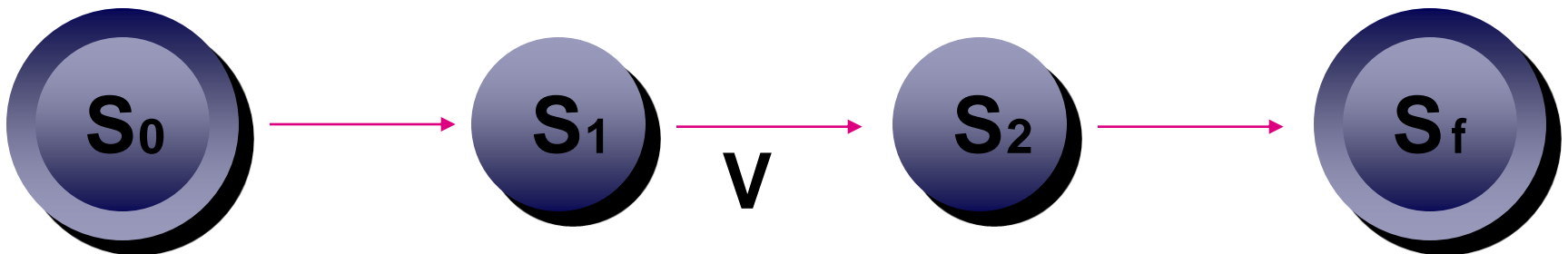
La **verifica dei vincoli** può essere:

a **immediate** (immediata):

la loro violazione annulla l'ultima modifica

b **deferred** (differita):

la loro violazione annulla l'intera applicazione



# Modifica dinamica del significato dei vincoli

- Ogni vincolo è definito di un tipo (normalmente "immediate")
- L'applicazione può modificare il tipo iniziale dei vincoli:
  - **set constraints immediate**
  - **set constraints deferred**
- Tutti i vincoli vengono comunque verificati, prima o poi

# Procedure

- **Moduli di programma che svolgono una specifica attività di manipolazione dei dati**
- **Non standard in SQL2 ma presenti nei principali sistemi relazionali**
- **Due momenti:**
  - **dichiarazione (DDL)**
  - **invocazione (DML)**
- **Con architettura client-server sono:**
  - **invocate dai client**
  - **memorizzate e eseguite presso i server**

# Esempio : prelievo dal magazzino

**magazzino**

<b>COD-PROD</b>	<b>QTA-DISP</b>	<b>QTA-RIORD</b>
<b>1</b>	<b>150</b>	<b>100</b>
<b>3</b>	<b>130</b>	<b>80</b>
<b>4</b>	<b>170</b>	<b>50</b>
<b>5</b>	<b>500</b>	<b>150</b>

**riordino**

<b>COD-PROD</b>	<b>DATA</b>	<b>QTA-ORD</b>

SQL: vincoli, trigger

# Specifica

- **L'utente indica un prelievo dando il codice del prodotto e la quantità da prelevare**
- **Se la quantità disponibile in magazzino non è sufficiente la procedura si arresta con una eccezione**
- **Viene eseguito il prelievo, modificando la quantità disponibile in magazzino**
- **Se la quantità disponibile in magazzino è inferiore alla quantità di riordino si predispose un nuovo ordine d'acquisto.**

# Interfaccia

```
PROCEDURE PRELIEVO  
  ( PROD      INTEGER ,  
    QUANT     INTEGER )
```

# Invocazione

```
PRELIEVO(4,150)
```

# Stato iniziale nella base di dati

COD-PROD	QTA-DISP	QTA-RIORD
4	170	50

# Realizzazione della procedura

- 1. DICHIARAZIONE VARIABILI**
- 2. LETTURA DELLO STATO**
- 3. SE LA QUANTITÀ DISPONIBILE È INSUFFICIENTE: ECCEZIONE**
- 4. AGGIORNAMENTO DELLO STATO**
- 5. SE LA NUOVA QUANTITÀ DISPONIBILE È INFERIORE ALLA QUANTITÀ DI RIORDINO: EMISSIONE DI UN ORDINE**

# Procedura

```
PROCEDURE PRELIEVO (PROD INTEGER, QUANT INTEGER) IS
Q1, Q2 INTEGER
X EXCEPTION
BEGIN
    SELECT QTA-DISP, QTA-RIORD INTO Q1, Q2
        FROM MAGAZZINO WHERE COD-PROD = PROD;
    IF Q1 < QUANT THEN RAISE(X);
    UPDATE MAGAZZINO
        SET QTA-DISP = QTA-DISP - QUANT
        WHERE COD-PROD = PROD;
    IF Q1 - QUANT < Q2 THEN INSERT INTO RIORDINO
        VALUES(PROD, SYSDATE, Q2)
END
```

# Esempio di invocazione

**PRELIEVO(4,150)**

**PROD=4, QUANT=150**

**SELECT QTA-DISP, QTA-RIORD INTO Q1, Q2  
FROM MAGAZZINO  
WHERE COD-PROD = PROD;**

<b>COD-PROD</b>	<b>QTA-DISP</b>	<b>QTA-RIORD</b>
<b>4</b>	<b>170</b>	<b>50</b>

**Q1 = 170, Q2 = 50**

# Invocazione (continua)

```
IF Q1 < QUANT THEN RAISE(X) non scatta  
UPDATE MAGAZZINO  
SET QTA-DISP = QTA-DISP - QUANT  
WHERE COD-PROD = PROD
```

COD-PROD	QTA-DISP	QTA-RIORD
4	20	50

Q1 - QUANT < Q2 è vero:

```
INSERT INTO RIORDINO  
VALUES(PROD, SYSDATE, Q2)
```

COD-PROD	DATA	QTA-RIORD
4	1997-10-10	50

# Regole attive (trigger)

- Moduli di programma che svolgono una specifica attività di manipolazione dei dati
- Non standard in SQL2 ma presenti nei principali sistemi relazionali
- Simili alle procedure, ma l'invocazione è automatica
- Seguono il paradigma **ECA**  
( **EVENTO-CONDIZIONE-AZIONE** )

# Paradigma ECA

- **evento:** modifica alla base di dati
- **condizione:** predicato
- **azione:** modifica alla base di dati, segnalazioni agli utenti
- **informalmente:**
  - quando accade l'evento
  - se la condizione è vera
  - allora si esegue l'azione

# Esempio: gestione automatica del riordino

## 1. EVENTO:

**UPDATE(QDISP) IN MAGAZZINO**

**2. CONDIZIONE: LA NUOVA QUANTITÀ DISPONIBILE È INFERIORE ALLA (NUOVA) QUANTITÀ DI RIORDINO:  
NEW.Q-DISP < NEW.Q-RIORD**

## 3. AZIONE:

- SE LA QUANTITÀ DISPONIBILE È INSUFFICIENTE: ECCEZIONE**
- EMISSIONE DI UN ORDINE**

# Regola attiva (trigger)

```
CREATE TRIGGER GESTIONE-RIORDINO
AFTER UPDATE OF QTA-DISP ON MAGAZZINO
WHEN (NEW.QTA-DISP < NEW.QTA-RIORD)
FOR EACH ROW
X EXCEPTION
BEGIN
    IF NEW.QTA-DISP < 0 THEN RAISE(X);
    INSERT INTO RIORDINO
    VALUES(NEW.COD-PROD, SYSDATE,
            NEW.QTA-RIORD)
END
```

# Esecuzione dell'applicazione

**UPDATE MAGAZZINO**

**SET QTA-DISP = QTA-DISP - 150**

**WHERE COD-PROD = 4**

<b>COD-PROD</b>	<b>QTA-DISP</b>	<b>QTA-RIORD</b>
<b>4</b>	<b>20</b>	<b>50</b>

# Esecuzione del trigger

**evento** : UPDATE(QTA-DISP) ON MAGAZZINO

**condizione** : VERA

**azione** : IF NEW.QTA-DISP < 0 THEN RAISE(X)

non scatta

**INSERT INTO RIORDINO VALUES  
(NEW.COD-PROD, SYSDATE,  
NEW.QTA-RIORD)**

<b>COD-PROD</b>	<b>DATA</b>	<b>QTA</b>
<b>4</b>	<b>1997-10-10</b>	<b>50</b>

# Problemi di progetto per procedure e trigger

- **Decomposizione modulare delle applicazioni**
- **Paradigma di invocazione:**
  - **esplicita (procedure)**
  - **implicita (trigger)**
- **Aumento di:**
  - **efficienza**
  - **controllo**
  - **riuso**

# Conseguenze dell'uso di procedure e trigger

- **Aumenta la responsabilità dell'amministratore della base di dati (rispetto al programmatore applicativo)**
- **Si sposta “conoscenza” dalle applicazioni allo schema della base di dati (independenza di conoscenza)**

# Esercizi

- **Riprendere le basi di dati per la gestione degli ordini ed esprimere:**
  - **un vincolo di integrità che impedisce la presenza di più di 100 dettagli per ciascun ordine.**
  - **una procedura che elimina tutti gli ordini e i relativi dettagli di un particolare cliente**
  - **un trigger che scatta quando viene cancellato un cliente ed elimina tutti gli ordini e i relativi dettagli di quel cliente**

# Un breve ripasso

- **MODELLO RELAZIONALE**
- **ALGEBRA RELAZIONALE**
  - SELEZIONE, PROIEZIONE, UNIONE, DIFFERENZA, JOIN
- **SQL - DATA DEFINITION**
  - CREATE DOMAIN, TABLE, INDEX, VIEW, ASSERTION, PROCEDURE, TRIGGER
  - DROP, ALTER
  - GRANT, REVOKE
- **SQL - DATA MANIPULATION**
  - SELECT
  - INSERT, UPDATE, DELETE