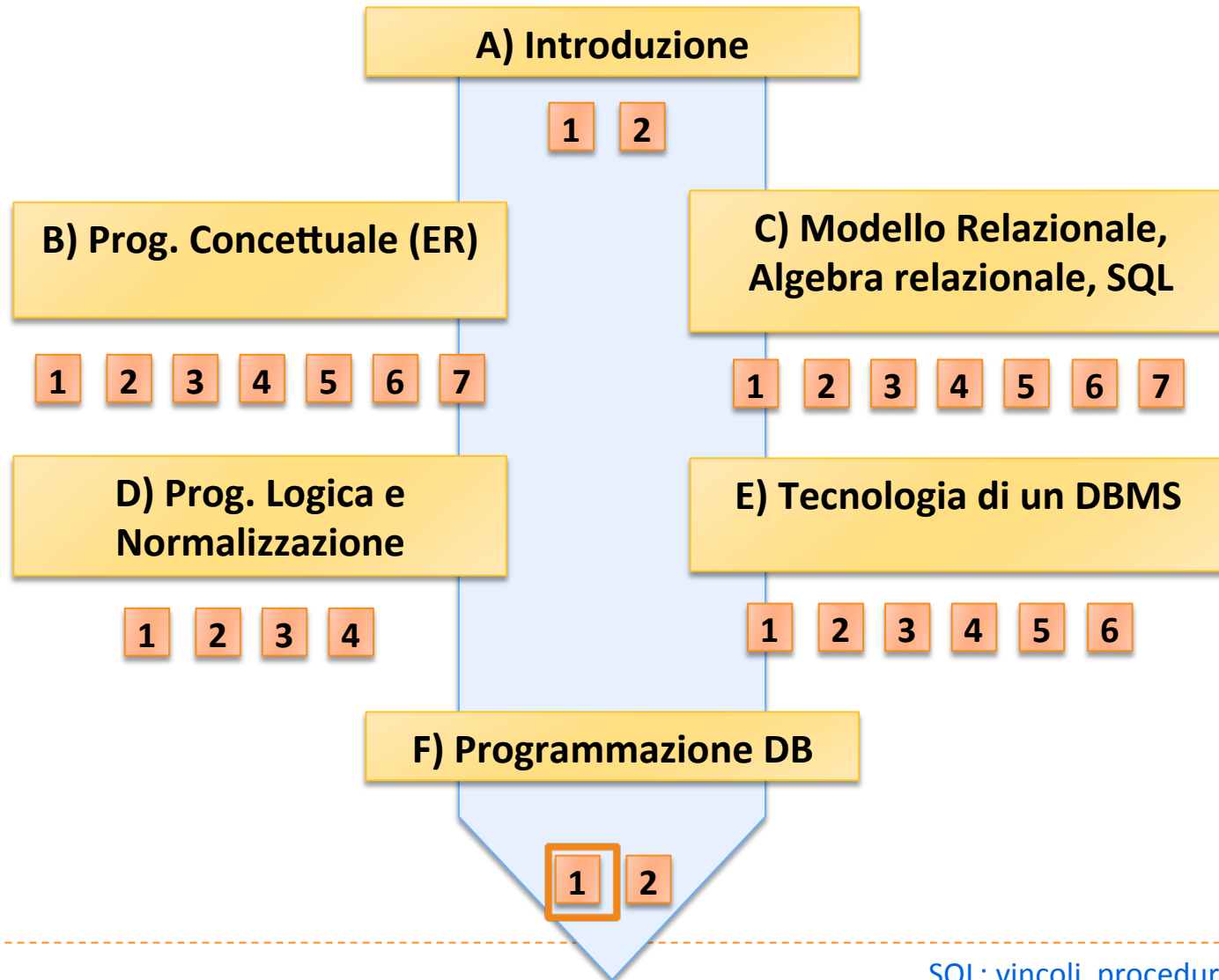


Basi di Dati

Vincoli, procedure e regole attive in SQL

Basi di Dati – Dove ci troviamo?



Indice

- ▶ Vincoli di Integrità
- ▶ Procedure (Stored Procedure)
- ▶ Regole attive (Trigger)

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

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

riordino

COD-PROD	DATA	QTA-ORD

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)
)
```

Assertzioni

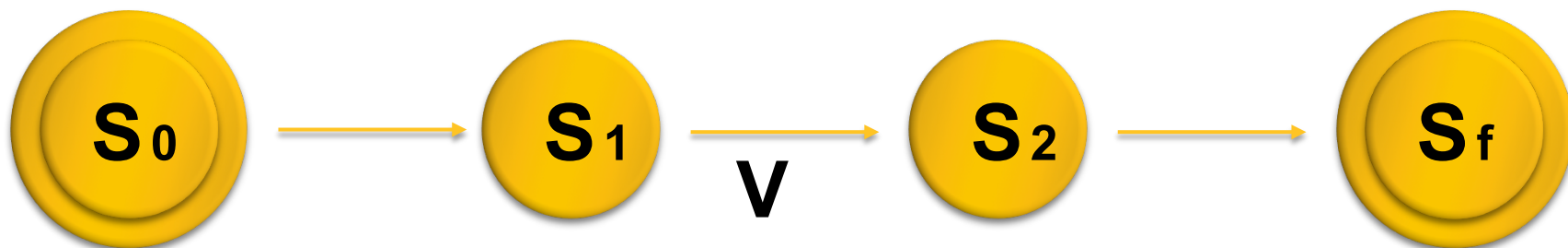
- ▶ Predicati espressi separatamente dalla definizione delle tabelle, che devono essere veri se valutati su istanze corrette (legali)

CREATE ASSERTION Ordini-Limitati **AS**

```
CHECK( 1000 >=
        SELECT COUNT(COD-PROD)
        FROM RIORDINO )
```


Significato dei vincoli

- ▶ La verifica dei vincoli può essere:
 - ▶ **Immediate** (immediata)
 - ▶ la loro violazione annulla l'ultima modifica
 - ▶ **Deferred** (differita)
 - ▶ la loro violazione annulla l'intera transazione



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 SQL-2 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

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

riordino

COD-PROD	DATA	QTA-ORD

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 predispone un nuovo ordine d'acquisto.

Procedura

- ▶ **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;
```

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

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	2012-10-10	50

Regole attive (trigger)

- ▶ Moduli di programma che svolgono una specifica attività di manipolazione dei dati
- ▶ Non standard in SQL-2 ma presenti nei principali sistemi relazionali
- ▶ Simili alle procedure, ma l'invocazione è automatica
- ▶ Seguono il paradigma
 - ▶ EVENTO-CONDIZIONE-AZIONE

Paradigma evento - condizione - azione (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

Tipologie di trigger

- ▶ **Row-level trigger**: Il trigger viene eseguito una volta per ciascuna tupla (row) della tabella coinvolta dall'evento di triggering. Comando per specificare un trigger row-level:
FOR EACH ROW
- ▶ **Statement-level trigger**: il trigger esegue soltanto una volta per evento, non separatamente per ogni tupla coinvolta

Trigger: variabili speciali

- ▶ In un trigger, il DBMS ci mette a disposizione una serie di variabili speciali, il cui valore è automaticamente assegnato al momento dell'invocazione
 - ▶ **NEW (tipo RECORD)**: contiene la nuova tupla per le operazioni di INSERT/UPDATE (row-level trigger), è NULL per operazioni di DELETE e per statement-level trigger
 - ▶ **OLD (tipo RECORD)**: contiene la vecchia tupla per le operazioni di UPDATE/DELETE (row-level trigger), è NULL per operazioni di INSERT e per statement-level trigger
 - ▶ Diverse altre variabili utili: consultare la documentazione del DBMS

Esempio: gestione automatica del riordino

- ▶ **EVENTO:**

- ▶ UPDATE(QDISP) IN MAGAZZINO

- ▶ **CONDIZIONE:**

- ▶ LA NUOVA QUANTITÀ DISPONIBILE È INFERIORE ALLA (NUOVA) QUANTITÀ DI RIORDINO: $NEW.Q-DISP < NEW.Q-RIORD$

- ▶ **AZIONE:**

- ▶ SE LA QUANTITÀ DISPONIBILE E' 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 = PROD

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

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)

COD-PROD	DATA	QTA
4	2012-10-10	50

Problemi di progetto per procedure e trigger

- ▶ Decomposizione modulare delle applicazioni
- ▶ Paradigma di invocazione:
 - ▶ esplicita (procedure)
 - ▶ implicita (trigger)
- ▶ Aumento di:
 - ▶ Efficienza
 - ▶ Controllo
 - ▶ Riutilizzo

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 (indipendenza 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