Basi di Dati Attive

Basi di dati attive
Definizione

- Una Base di Dati si dice attiva se dispone di un sottosistema integrato per definire e gestire regole di produzione (regole attive).

Basi di dati attive
Caratteristiche

- Le regole seguono il paradigma ECA (Event - Condition - Action)
- Il "processore delle regole" (rule engine) determina l'alternarsi tra l'esecuzione delle transazioni e quello delle regole (reattivo)
- Regole attive => Knoweledge Independence

Basi di dati attive

- Quasi tutti i DBMS relazionali sono Basi di dati attive ==> Trigger
- Difformità nei comportamenti dei vari Trigger
- Evoluzione estrema delle Basi di dati attive ==> sistemi per la gestione di Stream di Dati

Basi di dati attive
Trigger

- Fanno parte del DDL
- Seguono paradigma ECA
  - Event => primitive DML
  - Condition => predicato booleano in SQL
  - Action => sequenza di primitive SQL
- Si riferiscono (in genere) a una tabella (target)

Basi di dati attive
Trigger

- Due livelli di granularità
  - di tupla (row-level) => attivazione avviene per ogni tupla coinvolta
  - di primitiva (statement-level) => attivazione una sola volta per ogni primitiva SQL
- Modalità immediata o differita
- I trigger possono attivarsi in cascata l'uno con l'altro

Basi di dati attive
Definizione Trigger

- Ogni trigger è attivato da un solo evento.

```
create
    [Definer = {user|CURRENT_USER}]
trigger <trigger name> <trigger time>
            <trigger event>
on <table name> for each row <trigger
                                statement>
```

Basi di dati attive
Esempi Trigger

```
drop trigger limitaaumenti1;
create trigger limitaaumenti1
        after update on emp for each row
        update emp
                set sal=sal*1.2
                where sal>sal*1.2;

update emp
        set sal=1000 where ename ="Smith";
```

- Aggiorna 1000 a Smith anche se viola le regole, perchè ?

Basi di dati attive
Esempi Trigger

```
drop trigger limitaaumenti1;
create trigger limitaaumenti1
        before update on emp for each row
        update emp
                set sal=sal*1.2
                where sal>sal*1.2;

update emp
        set sal=1000 where ename ="Smith";
```

- Provare a fare doppio update. Che succede ?
- Perchè ?

Basi di dati attive
Esempi Trigger (corretto)

```
delimiter //
drop trigger limitaaumenti1;
create trigger limitaaumenti1
        before update on emp for each row
        begin
                if NEW.sal>OLD.sal*1.2
                then set NEW.sal=OLD.sal*1.2;
                end if;
        end; //
delimiter ;
```

Basi di dati attive
Esempi Trigger (corretto)

*************************** 2. row ***************************
Trigger: mytrig
Event: INSERT
Table: emp
Statement: set NEW.data=curdate()
Timing: BEFORE
Created: NULL
sql_mode:
Definer: root@localhost
character_set_client: utf8
collation_connection: utf8_general_ci
Database Collation: latin1_swedish_ci

Basi di dati attive
Trigger

Comprendere il comportamento collettivo dei trigger è più complesso del progettarne uno.

Basi di dati attive
Constraint

Alternativa ai trigger su attributi semplici

- Constraint come foreign key (già visti)

- Constraint espliciti usano la clausola check

create table prova (chiave int primary key,
(coinstraint ck_chiave) check(chiave>100));

Basi di dati attive
Applicazioni Trigger

- Il gestore dei trigger opera come sottosistema del DBMS
- *I trigger si occupano di gestire i vincoli di integrità, calcolare dati derivati, gestire dati replicati, gestire il versioning, la sicurezza e la privatezza dei dati, il logging delle azioni e la registrazione degli eventi.*

- Le regole aziendali applicate al DB favoriscono l'"indipendenza della conoscenza".

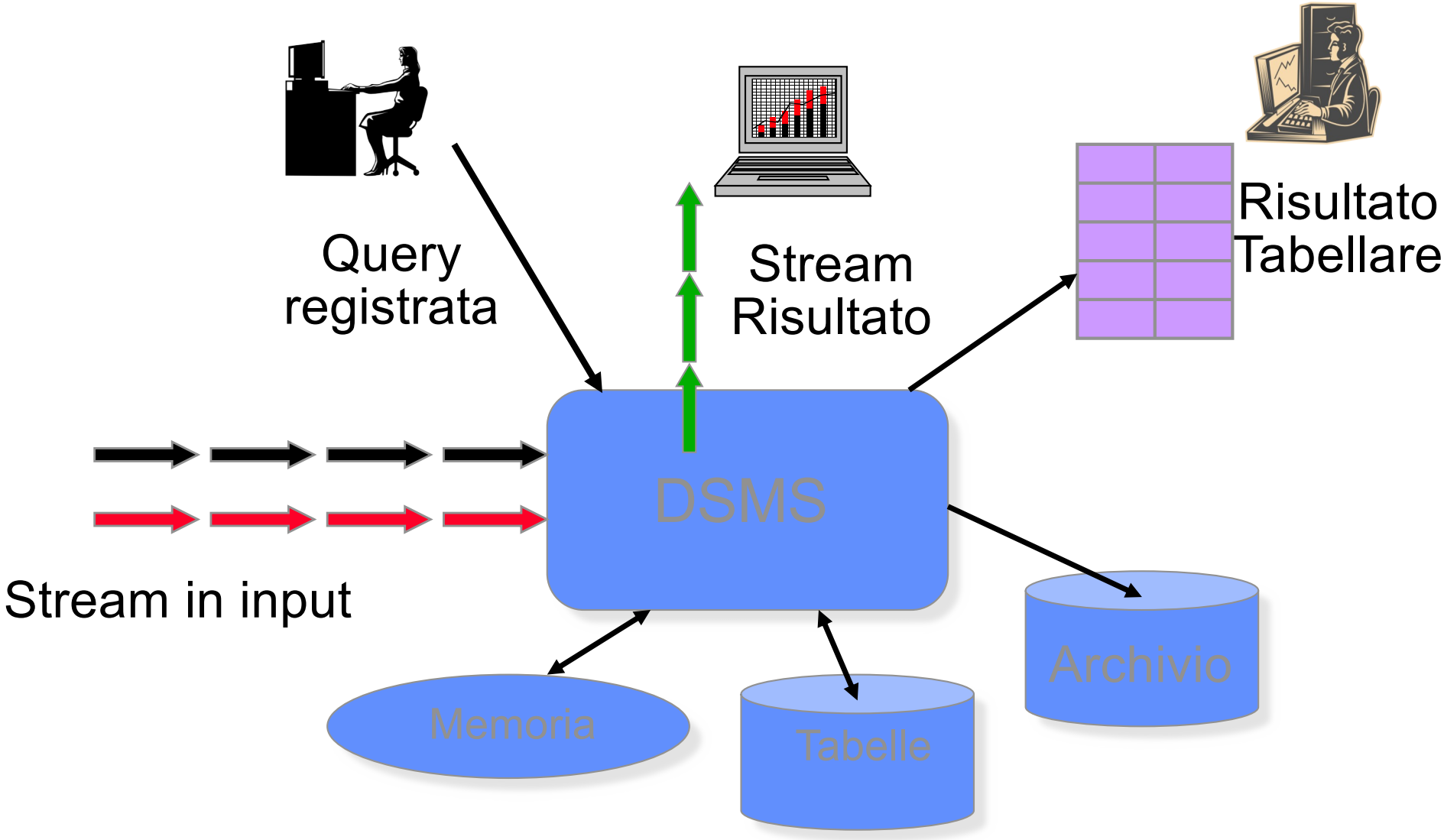
Basi di dati attive
Esempi Trigger (corretto)

```
delimiter //
drop trigger tropposal;
create trigger tropposal
        before update on emp for each row
        begin
                if (NEW.sal>(select min(sal) from emp
                        where deptno=NEW.deptno and job="manager"))
                then set @errore="salario troppo alto";
                end if;
        end; //
delimiter ;
```

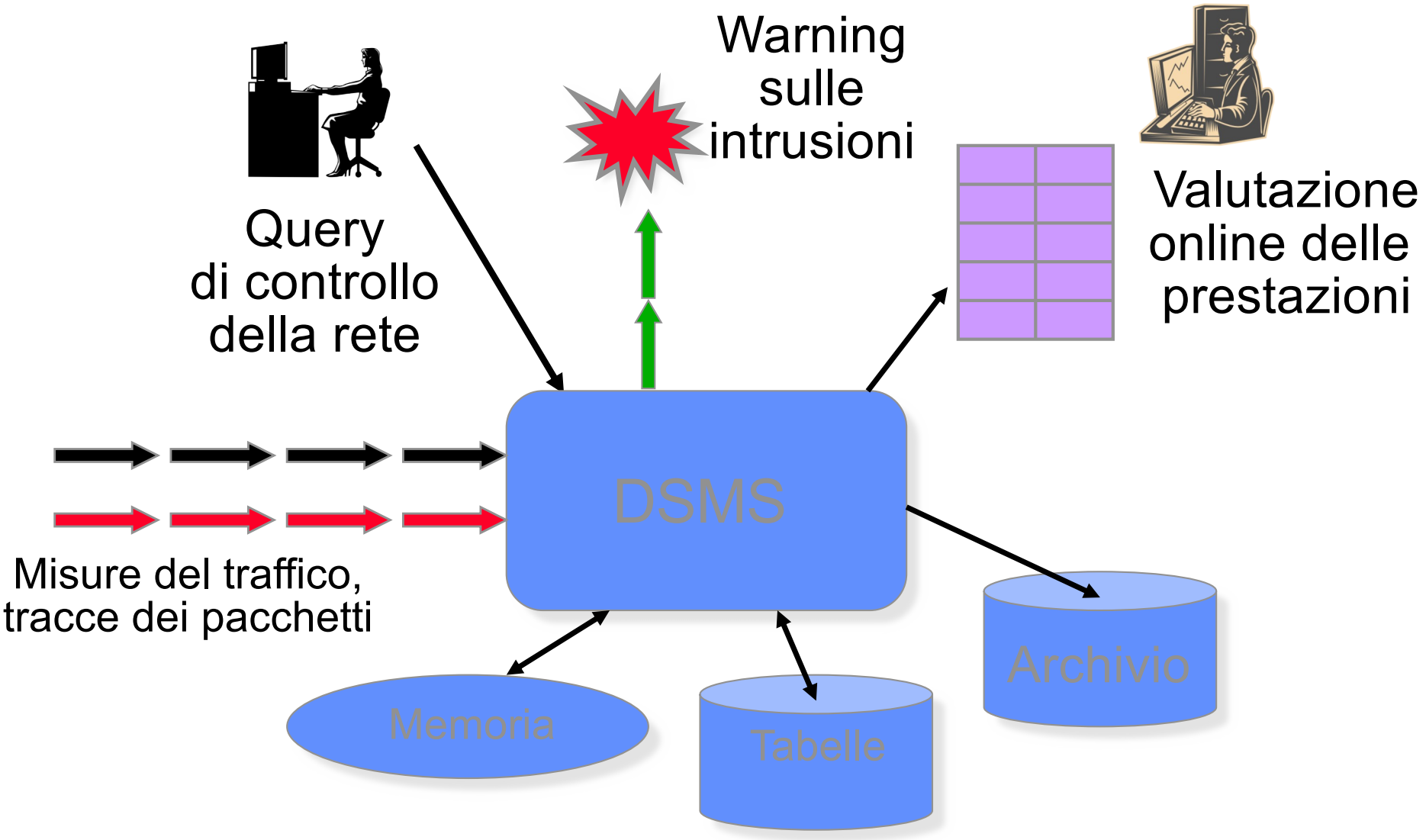
Data Streams

- DBMS Convenzionali: i dati costituiscono collezioni finite e presistenti
- In molte applicazioni: i dati sono streams continui, rapidi, infiniti, dipendenti-dal tempo
  - Controllo del traffico in rete
  - Sicurezza
  - Gestione delle chiamate telefoniche
  - Applicazioni finanziarie
  - Log del web e analisi di click-streams
  - Dati estratti da reti di sensori

L'idea



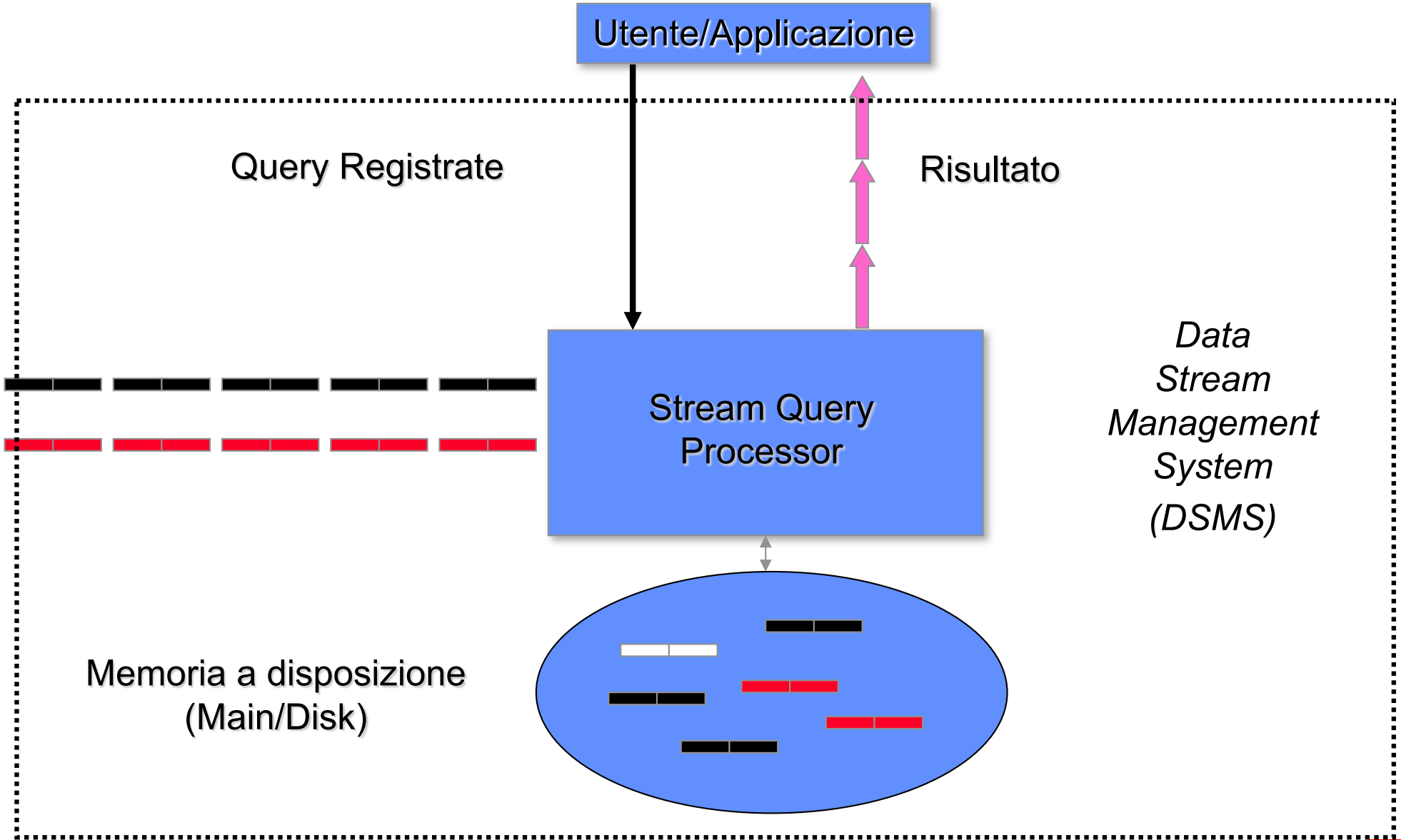
Applicazione tipo



Quale la relazione con i database attivi

- Nei database attivi: l'evento è una query, il sistema di trigger svolge la reazione
- In un stream database: l'evento è un flusso di dati, il sistema di gestione dello stream svolge la reazione
  - I sistemi per gestire data streams controllano in modo massivo molteplici eventi data-driven

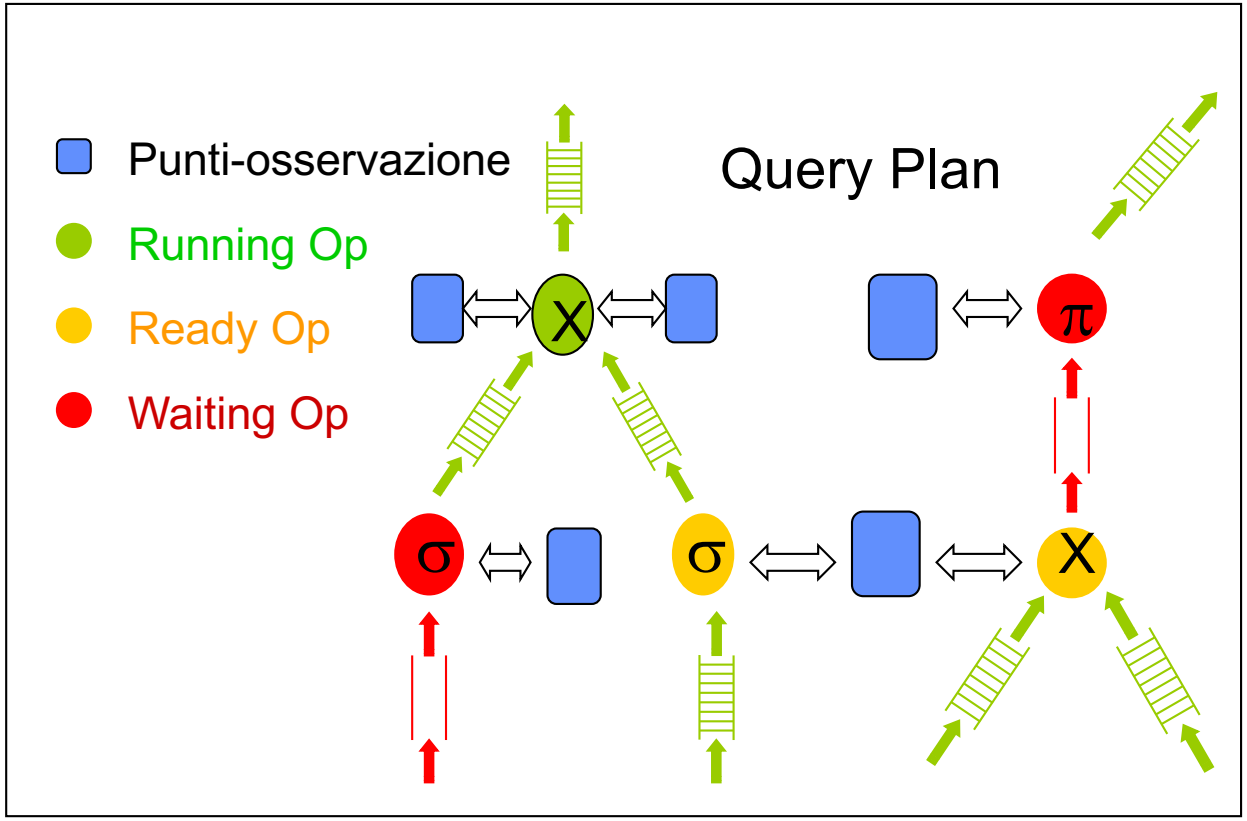
Architettura



Gestione degli stream

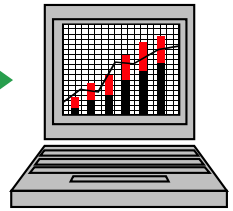
- Come esprimere query
  - Linguaggi ispirati a SQL
  - Linguaggi ispirati ai dataflow (grafici)
- Tempo di registrazione delle query
  - Predefinito
  - Ad-hoc
- Problemi semantici
  - Operatori “blocking”: *aggregazioni, order-by*
  - Streams interpretati come insiemi o liste
- Ottimizzazione multi-query
  - Trovare le parti comuni delle query
  - E’ possibile perché le query non cambiano

Architettura



Input
Data Streams

Output
stream



Le applicazioni registrano query continue

Gli utenti fanno query continue e ad-hoc

Gli amministratori controllano l'esecuzione tramite parametri a run-time

Finestra dati

- Porzione di stream, per rendere finito uno stream (potenzialmente infinito)
- Le finestre sono definite tramite: tempo iniziale e finale, numero di tuple, o tramite valori di breakpoint
- Finestre sullo stesso stream possono avere una intersezione non vuota

Problemi di valutazione di query: Approssimazione

- Perchè l'approssimazione?
  - Gli streams arrivano troppo velocemente
  - Risposte esatte richiedono memoria illimitata o risorse computazionali eccessive
  - Quando è coinvolta l'intera storia
- Problemi con l'approssimazione
  - Come contollarla?
  - Come farla capire all'utente?
- C'e' un trade-off tra accuratezza-efficienza-memoria

Problemi di valutazione di query: Adattatività

- Perchè la adattatività?
  - Le queries restano presenti a lungo, ma possono cambiare le loro caratteristiche (p.e.: il loro carico)
  - L'arrivo degli stream può avere fluttuazioni importanti
- Problemi di adattività
  - Allocazione adattativa delle risorse (memoria, calcolo)
  - Piani di esecuzioni adattativi delle query

Applicazioni tipiche/1

- **Gestione della rete e del traffico** (p.e., Sprint)
  - Streams di misure e tracce di pacchetti
  - Query: trovare anomalie, modificare il routing
- **Sicurezza delle reti informatiche** (e.g., iPolicy, NetForensics/Cisco, Netscreen)
  - Streams di richieste di risorse
  - Query: filtro di URL, ricerca intrusioni, attacchi e viruses
- Dati sulle **chiamate telefoniche** (e.g., AT&T)
  - Streams di record relativi alle chiamate
  - Query: scoperta di frodi, ricerca di pattern ricorrenti tra le chiamate

Applicazioni tipiche/2

- **Applicazioni finanziarie** (e.g., Traderbot)
  - Stream di dati di titoloi e trading
  - Query: ricerca di opportunità finanziarie
- **Web tracking e personalizzazione** (e.g., Yahoo, Google, Akamai)
  - Streams di click e di record di log
  - Query: monitoring, analisi, personalizzazione
- **Dati massivi** (e.g., Astronomy Archives)
  - Stream che può venir analizzato solo “durante” il flusso di dati
  - Query: operano al meglio, caso per caso

Basi di dati attive
Applicazioni

- Procedure e trigger permettono di realizzare buona parte dell'applicazione all'interno del DB stesso.

- Problema analogo è avere un'applicazione, in un linguaggio in alto livello, che deve interagire con l'SQL (Cursori e CLI).

Basi di dati attive
Stored Procedure

- Insieme di istruzioni SQL, memorizzate nel server, richiamate al bisogno.

create procedure nome([parametro,…])

[SQL security{definer|invoker}]

istruzioni//parametri:

[IN | OUT | INOUT] nomepar tipo;

Basi di dati attive
esempio Stored Procedure

```
create procedure contajob (IN job
    varchar(20), OUT conta int)
        select count(*) into conta from emp
            where emp.job=job;
```

Basi di dati attive
esempio Stored Procedure

- Per vedere la procedura
  - *show create procedure contajob\G*
- Per richiamarla
  - *call contajob('clerck',@a);*
- Per verificarla
  - *select @a;*

Basi di dati attive
Stored Procedure

- Per vedere tutte le stored procedure create
  - *show procedure status;*

Basi di dati attive
Cursor

- Un cursore è uno strumento che permette ad un programma di accedere alle righe di una tabella, una alla volta ➔ query scalari

- Un cursore può essere definito all'interno di un qualsiasi programma ad alto livello

Basi di dati attive
esempio di Cursor

- Vedremo un esempio di procedura per contare (par3) tutti gli impiegati di un certo dipartimento (par2) che hanno lo stipendio maggiore di un certo numero (par1)

Procedure: selsaldip

sql_mode:

```
delimiter //
Create  PROCEDURE selsaldip(par1 int, par2 char(2), out par3 int)
begin
declare finito int default 0;
declare a int;
declare b varchar(20);
declare selsaldip cursor for select sal,ename from emp where deptno=par2;
declare continue handler for sqlstate '02000'
                set finito=1;
open selsaldip;
set par3 =0;
fetch selsaldip into a,b;
ciclo: while not finito do
                        if a>par1 then
                                set par3=par3+1;
                                fetch selsaldip into a,b;
                        else
                                fetch selsaldip into a,b;
                        end if;
                end while ciclo;
end;//
delimiter ;
```

Basi di dati attive
Stored Function

- Simili alle stored procedure, restituiscono un solo valore

create function nome([parametro,…])

returns tipo

[SQL security{definer|invoker}]

istruzioni//parametri: nomepar tipo;

Basi di dati attive
Call Level Interface

- Insieme di funzioni che permettono di interagire con il Dbms
- Fasi :
  - Connessione
  - Comando SQL con richiesta
  - Risposta in strutture
  - Chiusura connessione

Basi di dati attive in MySQL

DB attive in MySQL caratteristiche

- Query cache
  - Risultati di query già effettuate
  - Trasparente all'utente
  - Salta parser ed optimizer
  - Tiene traccia delle tabelle modificate
  - Il transazionale ne limita l'utilizzo
    - A causa dell'MVCC, InnoDB ci accede in maniera complessa

DB attive in MySQL caratteristiche

- Introdotte dalla 5.0 e 5.1 in poi
- Le stored code usano una speciale estensione dell'SQL (SQL/PSM) che permette strutture procedurali, loop e condizioni
- Stored procedure e function accettano parametri e tornano risultati, trigger ed eventi no.

DB attive in MySQL
pro e contro

- Vantaggi
  - Sicurezza
  - Riduzione latenza
  - riutilizzo
- Svantaggi
  - Complessità di calcolo maggiore
  - No debugging
  - Minor controllo sulle risorse (dati)

DB attive in MySQL
Stored procedure e stored function

- L'ottimizzatore non riesce a stimare il loro costo
- Spesso più veloci di alcune query
- È possibile usare una procedura di un altro DB
  - *Call DB.procedure;*

DB attive in MySQL
esempio Stored Procedure

```
drop procedure if exists inserisci_righe; delimiter //
create procedure inserisci_righe (IN loops int)
Begin
    declare v1 int;
    set v1=loops;
    while v1 > 0 do
        insert into test_table values(NULL,0,'abcd','abcd');
        set v1=v1-1;
    end while;
end; //
delimiter;
```

DB attive in MySQL
Trigger

- Simulano coinstraint e/o foreign key su motori che non le supportano (MyIsam)
- Un solo trigger per tabella, ad ogni evento
- Solo livello riga (inefficiente per grandi Db)
- Pericoli :
  - Oscurano quello che il server sta realmente facendo
  - Difficili da debuggare
  - Possono causare deadlock

DB attive in MySQL
Eventi

- Nuova forma di stored code della 5.1
- Simili al ‘cron’, ma completamente interni al MySQL server
- Eseguiti da un event scheduler thread separato
- Né input, né output

create event fa_qualcosa on schedule every 1 week

do

call qualcosa_su_tabelle(‘miodb’);

DB attive in MySQL
Cursori

- Possono essere usati solo da dentro stored procedure e solo per leggere (anche se più di uno e nidificati)

- Falso senso di efficienza e pericolosità

DB attive in MySQL
Prepared Statement

Possibilità di costruire e usare statement già preparati, parametrizzabili e memorizzati

```
set @st from 'select sqrt(pow(?,2) + pow(?,2)) as ipotenusa';
prepare stmt1 from st;
set @a=3;
set @b=4;
execute stmt1 using @a, @b ;
```

DB attive in MySQL
User Defined Function

- A differenza delle stored function scritte in SQL, possono essere scritte in un qualsiasi linguaggio da compilare

- Tipicamente usate per inviare pacchetti sulla rete