Capitolo 4

Funzionalità del sistema DEMI

In questo capitolo vengono definite le caratteristiche e le funzionalità che devono essere implementate nel sistema DEMI.

Poiché il sistema non deve essere specifico all'ambiente HEP, in questa descrizione si utilizza una terminologia "generale", non propria dell'ambiente della fisica delle altre energie.

Ulteriori informazioni riguardo l'analisi e la specifica dei requisiti si possono trovare nello User Requirement Document [1].

4.1 Introduzione

I vari siti possono differire tra loro sotto diversi aspetti: nel tipo e nel numero di risorse computazionali, nel numero di utenti, ecc... In particolare nei vari siti si possono voler considerare diverse "politiche" di gestione delle risorse locali. Ad esempio, in certe realtà le workstation potrebbero essere dedicate alle elaborazioni DEMI, mentre in altri ambienti i calcolatori potrebbero essere utilizzati dal sistema DEMI solo se non utilizzati dagli altri utenti.

Per questi motivi è opportuno separare l'amministrazione del sistema in due livelli. In ogni sito la gestione è affidata ad un *local administrator*: suo compito è quello di gestire e controllare le attività locali del sistema DEMI (in particolare la gestione delle risorse). L'amministrazione del sistema globale è invece affidata ad un *global administrator*, responsabile nel gestire e coordinare l'intero sistema: il global administrator deve occuparsi di attività quali la configurazione dell'ambiente, la gestione dei database globali, ecc...

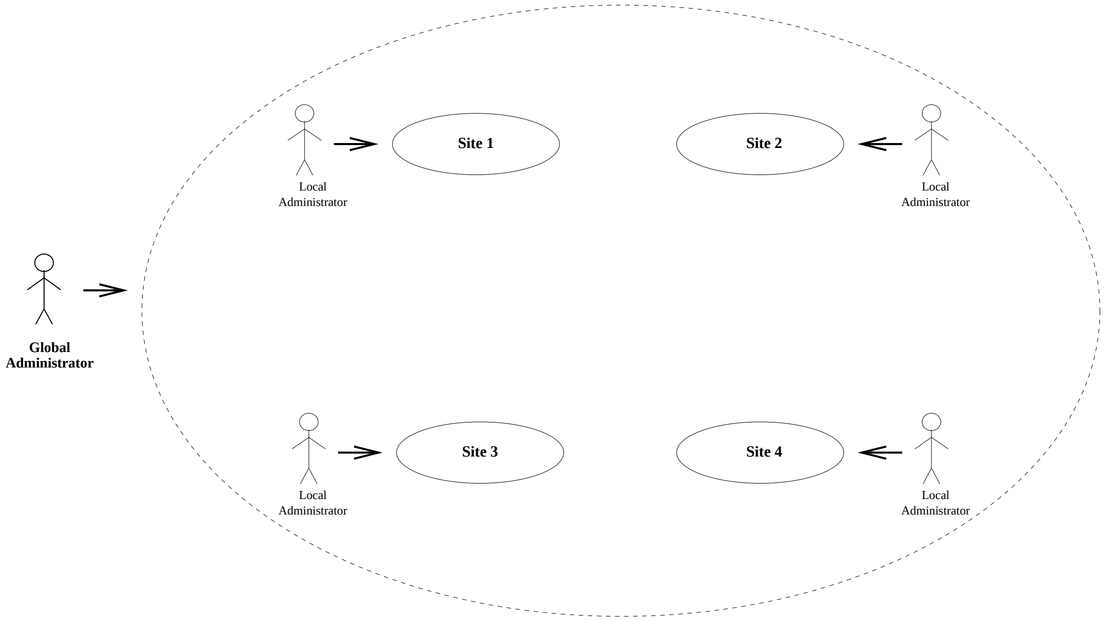


Figura 4.1: Global administrator e local administrator

4.2 Definizione dell'ambiente globale

Per definire l'ambiente globale in cui il sistema DEMI deve operare, è necessario specificare:

1. la configurazione del sistema;
2. le architetture dei calcolatori utilizzati;
3. i modelli di tape device usati.

4.2.1 Gestione della configurazione del sistema

Il global administrator è responsabile nella definizione della configurazione del sistema DEMI: devono essere specificati i siti componenti il sistema, e le loro caratteristiche.
Per ogni sito è necessario definire:

- il nome del sito (es. *Padova*);
- una descrizione (es. *Sito dell'INFN sezione di Padova*);
- un "indirizzo" che permetta di accedere al sito considerato.

Il global administrator deve essere in grado di:

- aggiungere al sistema nuovi siti;
- rimuovere siti dal sistema DEMI;
- modificare informazioni relative a siti precedentemente inseriti.

4.2.2 Gestione delle architetture dei computer

Come detto, i computer impiegati nel sistema DEMI possono essere di tipo e caratteristiche diverse. Questi calcolatori possono essere classificati a seconda della loro *architettura*: il tipo di architettura definisce la compatibilità binaria per i file eseguibili. Quindi due computer hanno la stessa architettura se e solo se per ogni task è possibile usare nei due sistemi gli stessi file eseguibili per l'esecuzione. Solitamente l'architettura è determinata dal modello del computer e dal sistema operativo considerato.

Le informazioni relative alle architetture dei calcolatori devono essere memorizzate in maniera persistente. A tal fine si può considerare un'apposita struttura dati, chiamata **Architecture Database**, in cui ogni elemento è definito dai seguenti campi:

- un identificatore (es. *alpha-win32*);
- una descrizione (es. *Computer con processore alpha, con sistema operativo Windows NT*);
- altre eventuali informazioni.

Il global administrator deve poter:

- definire nuove architetture di calcolatori;
- rimuovere elementi dall'Architecture Database;
- modificare informazioni relative ad architetture prima definite;
- visualizzare gli elementi definiti nell'Architecture Database.

4.2.3 Gestione dei modelli di tape device

I dischi in genere non sono sufficienti per la memorizzazione di tutti i dati. Si possono allora usare nastri, che hanno un'elevata capacità di memorizzazione, e costi relativamente contenuti. I nastri, e i relativi tape device (i dispositivi che permettono di scrivere/leggere dati sui/dai nastri) possono essere di tipo diverso (DLT, DAT, Exabyte 8mm, ecc...).

Il global administrator deve gestire attraverso un apposito database, chiamato **Tape Device Model Database**, i modelli di tape device (e dei relativi nastri) utilizzati nel sistema DEMI.
Ogni elemento di questo database viene definito dalle seguenti informazioni:

- un identificatore (es. *DLT*);
- una descrizione (es. *tape device DLT, per nastri TZ87*);
- la capacità di un nastro di questo modello, ovvero la massima quantità di dati che è possibile scrivere in un nastro di questo tipo (es. *30 GB*);
- altre eventuali informazioni.

Le operazioni che devono poter essere effettuate dal global administrator sono:

- la definizione di nuovi modelli di tape device;
- la cancellazione di elementi dal Tape Device Model Database;
- la modifica di informazioni relative ad elementi definiti nel database;
- la visualizzazione degli elementi definiti nel Tape Device Model Database.

Come rappresentato in figura 4.2, l'Architecture Database e il Tape Device Model Database sono due database globali, di cui esiste un'unica istanza per tutto il sistema DEMI. Questi database possono essere modificati solamente dal global administrator, mentre gli utenti dei vari siti (local administrator compresi) vi possono accedere solo per operazioni di lettura.

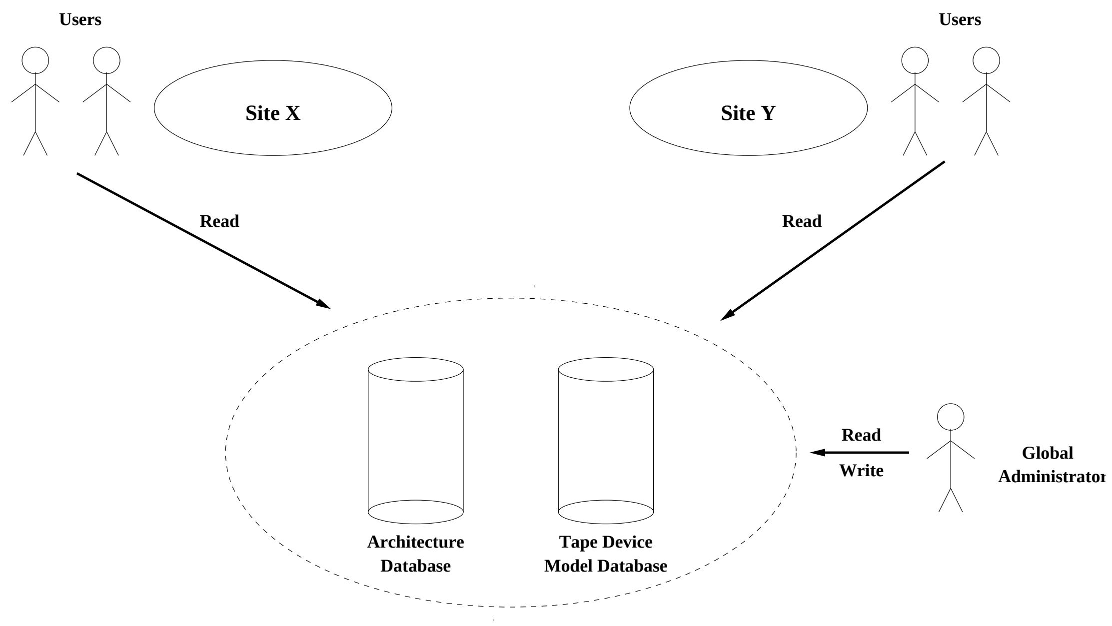


Figura 4.2: L'Architecture Database e il Tape Device Model Database

4.2.4 Global administrator

Possono esistere più global administrator, responsabili della gestione del sistema DEMI a livello globale. Le informazioni relative a questi amministratori devono essere memorizzate in un'apposita struttura dati, chiamata **Global Administrator Database**. Ogni elemento di questo database è definito dai seguenti campi:

- un identificatore, chiamato **user identifier** (es. *sgaravat*);
- la password (criptata) che l'administrator deve specificare quando vuole gestire il sistema DEMI;
- l'indirizzo e-mail dell'amministratore (es. *sgaravat@domain*);
- altre eventuali informazioni.

Su questo database possono essere fatte, da parte di un global administrator, le classiche operazioni di inserimento, cancellazione, modifica, ricerca e visualizzazione.

4.3 Definizione dell'ambiente locale

I local administrator sono responsabili dell'amministrazione dei corrispondenti siti, in particolare della gestione delle risorse locali.
Si devono considerare tre tipi di risorse:

- i computer, utilizzati per le elaborazioni sottomesse dagli utenti;
- i disk buffer, usati per la memorizzazione dei dati;
- i tape device, utilizzati per la lettura/scrittura dei dati da/su nastri.

In ogni sito è necessario memorizzare i dati relativi a queste tre classi di risorse, considerando tre specifiche strutture dati:

- il **Local Computer Database**;
- il **Local Buffer Database**;
- il **Local Tape Device Database**.

4.3.1 Gestione dei computer

Ogni elemento del Local Computer Database definisce un calcolatore presente nel sito, utilizzato dal sistema DEMI, e viene rappresentato da vari campi, quali:

- il nome o l'indirizzo IP del computer (es. *ntde01.pd.infn.it*);
- la descrizione (es. *alpha station 3000/600, con sistema operativo Windows NT Server 4.0*);
- l'architettura di questo computer (es. *alpha-win32*): questa architettura deve essere definita nell'Architecture Database;
- un valore numerico, chiamato 'performances', che definisce le prestazioni di questo computer: più alto è il valore di questo parametro, migliori sono le prestazioni. Il valore di questo campo può essere ottenuto attraverso un opportuno benchmark, e può quindi essere espresso, a seconda dei casi, in MIPS, MFLOPS, SPECINT, SPECFP, ecc... Nell'ambiente HEP si considera in genere il *CERN benchmark* [4], in cui si valutano le performance dei calcolatori considerando una serie di elaborazioni specifiche di fisica delle alte energie (l'unità di misura in questo caso è la *CERN Unit*).

Il local administrator deve essere in grado di definire nuovi computer nel Local Computer Database, di rimuovere calcolatori, di "disabilitare" computer (ovvero eliminare questi calcolatori dalla configurazione del sistema, senza però eliminarli dal database), di "riabilitare" computer, di modificare informazioni di un elemento presente nel Local Computer Database, di verificare lo stato dei vari calcolatori presenti nel sito, ecc...

4.3.2 Gestione dei disk buffer

I disk buffer vengono utilizzati dal sistema DEMI per la memorizzazione dei file di dati definiti nel Local Data Database (vedi par. 4.4.2), e dei file "temporanei" prodotti nei vari processing (dati che vengono cancellati quando non sono più necessari). Un disk buffer può essere un "intero" disco, o solamente la parte di un disco.

Si può supporre che esista un file system comune (implementato attraverso sistemi quali NFS [50], AFS [51], DCE/DFS [38], Samba [31], ecc...), per cui ogni computer del sito può accedere direttamente ad ogni buffer.
Ogni elemento del Local Buffer Database rappresenta uno di questi disk buffer, ed è descritto dai seguenti campi:

- un identificatore (es. *disk1*);
- il computer dove il disk buffer è fisicamente installato (es. *osde01.pd.infn.it*): questo elemento deve essere definito nel Local Computer Database;
- il pathname con cui si "accede" al buffer (es. */DEMI/disk1*);
- lo spazio totale disponibile in questo disk buffer per la memorizzazione di file DEMI (es. 2 Gbyte);
- altre eventuali informazioni.

Il local administrator deve essere in grado di aggiungere nuovi disk buffer al sistema DEMI, di rimuovere buffer dal sistema, di modificare informazioni di elementi presenti nel Local Buffer Database, di verificare lo stato dei vari buffer presenti nel sito (spazio usato, spazio disponibile, ecc...), ecc...

4.3.3 Gestione dei tape device

Il Local Tape Device Database è usato per registrare le informazioni relative ai tape device presenti nel sito, utilizzati dal sistema DEMI per la lettura/scrittura di file di dati da/su nastro. Ogni elemento di questo database è definito da varie informazioni quali:

- un identificatore (es. *tapedev1*);
- il corrispondente modello di tape device (es. *DLT*): questo modello deve essere definito nel Tape Device Model Database;
- il computer dove il device è fisicamente installato (es. *osde01.pd.infn.it*);
- il pathname con cui si "accede" al tape device (es. */dev/rmt0h*);
- altre eventuali informazioni.

Il local administrator deve poter essere in grado di aggiungere nuovi tape device al sistema DEMI, di rimuovere tape device dal sistema, di modificare informazioni di un elemento presente nel Local Tape Device Database, di rilevare lo stato dei vari tape device presenti nel sito, ecc...

Come descritto in figura 4.3, in ogni sito esistono specifiche istanze locali del Local Computer Database, del Local Buffer Database e del Local Tape Device Database. I local administrator possono accedere a questi database con operazioni di lettura-scrittura, mentre gli altri utenti del sito possono effettuare solo operazioni di lettura. I Local Computer Database e i Local Tape Device sono "in relazione" con i database globali Architecture Database e Tape Device Model Database, rispettivamente per le definizioni delle architetture dei computer e dei modelli di tape device.

4.3.4 Gestione degli utenti

In ogni sito del sistema DEMI deve essere utilizzato uno specifico database, chiamato **User Database**, per memorizzare le informazioni relative agli utenti DEMI del sito locale. Ogni utente deve essere definito nel database attraverso i seguenti campi:

- uno **user identifier** (es. *rossi*);
- la password (criptata) che l'utente deve specificare quando vuole "accedere" alle funzionalità offerte dal sistema DEMI;
- l'indirizzo e-mail dell'utente (es. *rossi@localdomain*);
- il "ruolo" di questo utente:

  la persona considerata può essere un utente generico, un local administrator, un operatore (responsabile del mount dei nastri negli appositi tape device). Un utente può avere anche più ruoli (ad esempio può essere allo stesso tempo un utente generico e un operatore);

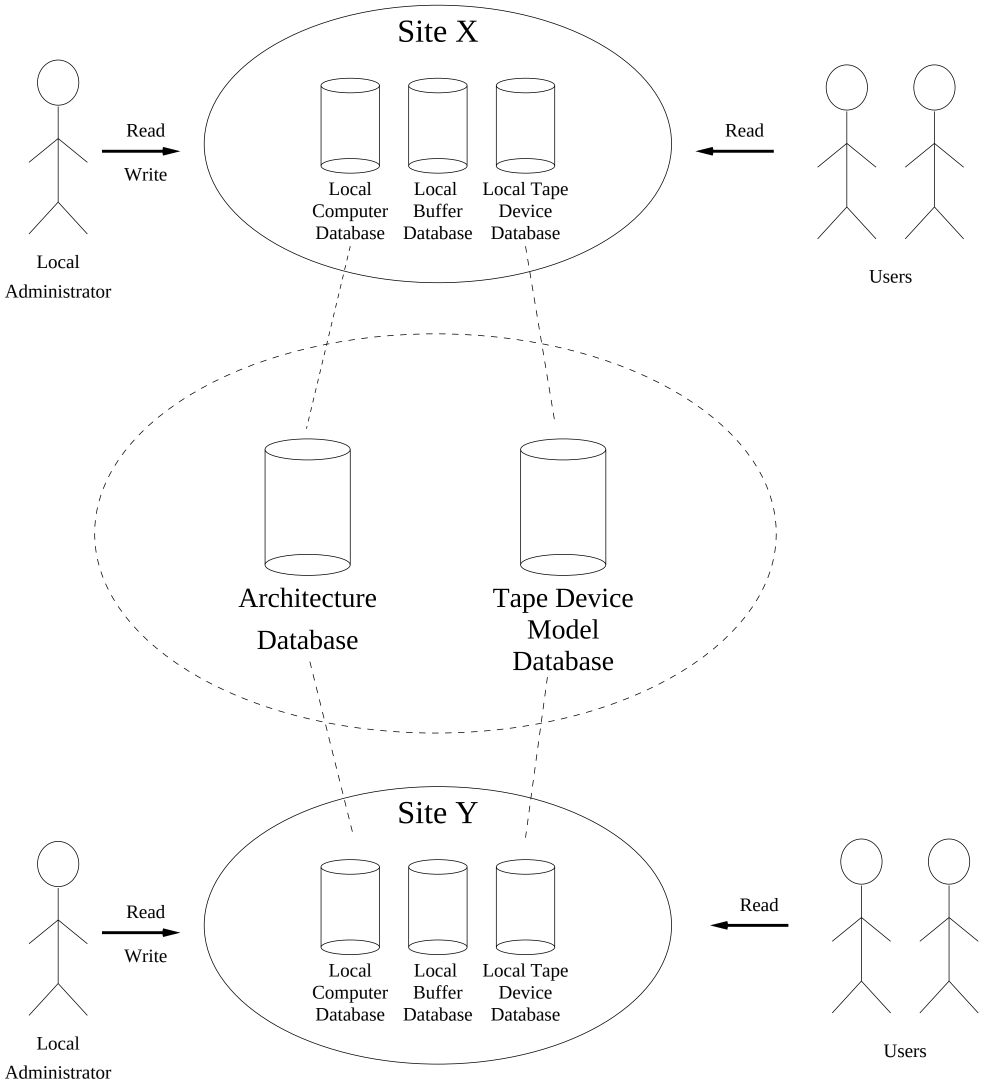


Figura 4.3: I database usati per la gestione delle risorse presenti nei vari siti

- altre eventuali informazioni.

Il local administrator può effettuare sullo User Database operazioni di inserimento, cancellazione, modifica, ricerca e visualizzazione.

4.4 Gestione dei dati

In generale, in ogni sito del sistema DEMI possono essere presenti file di dati, memorizzati su disk buffer (vedi par. 4.3.2) e/o su nastri (vedi par. 4.3.3).

Anche se i dati possono essere distribuiti geograficamente tra i vari siti, essi devono essere organizzati in un unico database logico, chiamato **Data Database**.

Per "accedere" ad un file di dati, l'utente non deve preoccuparsi di dove esso è fisicamente memorizzato, e non deve nemmeno conoscere il relativo pathname o identificatore del nastro: è necessario specificare semplicemente un "nome logico", chiamato **file identifier**, che permette di identificare senza ambiguità il file richiesto.

La struttura del sistema di gestione dei dati è rappresentata in figura 4.4: il Data Database consiste di un **Global Data Database** "centrale" e, in ogni sito del sistema DEMI, di un **Local Data Database**.

4.4.1 Global Data Database

Nel Global Data Database sono memorizzate le informazioni relative a tutti i file di dati gestiti dal sistema DEMI. Ognuno di essi viene descritto dai seguenti campi:

- il corrispondente file identifier;
- la descrizione del tipo di dati memorizzati nel file (es. *QQPS, 183 GeV, 97_1 MC*);
- il nome dei siti in cui il file è fisicamente presente (uno stesso file può essere presente in più siti);
- altre eventuali informazioni.

4.4.2 Local Data Database

In ogni sito del sistema DEMI è presente un Local Data Database, in cui vengono memorizzate informazioni relative ai file di dati fisicamente presenti nel sito locale. Ognuno di questi file è rappresentato nel database dai seguenti campi:

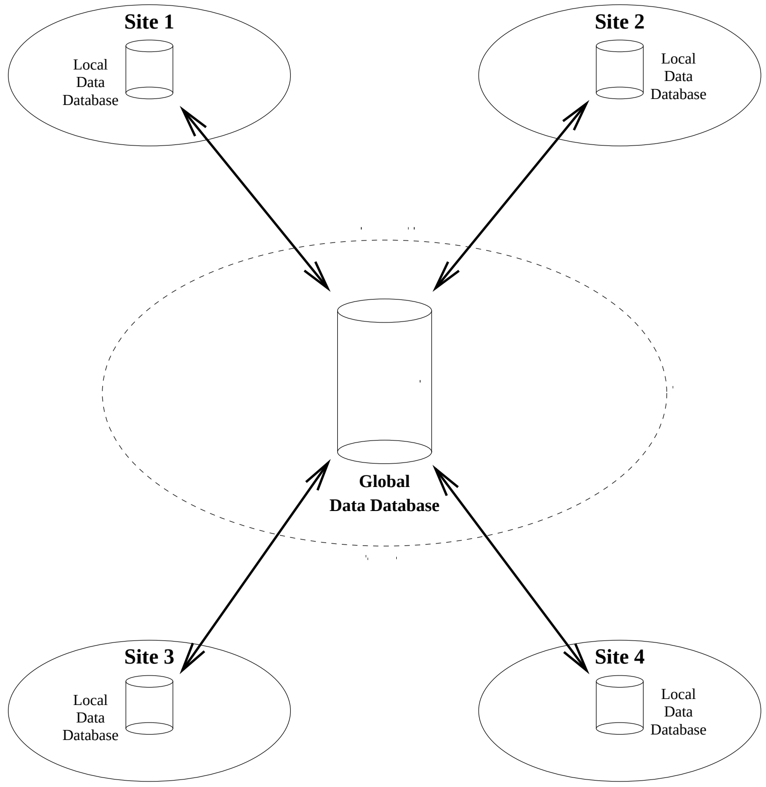


Figura 4.4: Data Database

- il file identifier;
- per ogni "occorrenza" di questo file nel sito (il file può essere presente in più dischi e/o nastri), sono specificate le seguenti informazioni:
  - il tipo di supporto di memoria di massa utilizzato per la memorizzazione del file, che può essere:
    - un disk buffer;
      in questo caso è necessario specificare l'identificatore di questo buffer (vedi par. 4.3.2);
    - un nastro;
      in questo caso è necessario indicare l'identificatore di questo nastro (vedi par. 4.4.3);
- altre eventuali informazioni.

Si supponga, ad esempio, che un certo file di dati sia presente nel sito *SiteX*, memorizzato su un disk buffer. Nel Global Data Database deve essere presente un elemento per questo file, che indica appunto che il file è presente nel sito *SiteX*. Nel Local Data Database del sito *SiteX*, deve essere definito un elemento per questo file, con l'indicazione che esso è memorizzato su un disk buffer, di cui viene dato l'identificatore.

Quindi nel Global Data Database vi è un elemento per ogni file di dati gestito dal sistema DEMI, mentre nel Local Data Database sono memorizzate informazioni solamente per quei file che sono presenti nel corrispondente sito.

Ogni utente del sistema DEMI deve poter interrogare il Data Database, utilizzando in queste query uno o più degli attributi descritti.

I vari campi del Data Database vengono aggiornati automaticamente dal sistema dopo ogni processing, per registrare i file di output prodotti: non devono essere gli utenti ad aggiornare manualmente il database. Ogni administrator deve comunque poter essere in grado di gestire direttamente il database, per inserire nuovi elementi, per modificare attributi di elementi presenti nel database, per cancellare elementi, ecc...

4.4.3 Gestione dei nastri

In ogni sito le informazioni relative ai nastri in cui vengono memorizzati i file di dati definiti nel Data Database, devono essere gestite attraverso un'opportuna struttura dati, chiamata **Local Tape Database**. In questo database ogni nastro è rappresentato dalle seguenti informazioni:

- un identificatore, chiamato **tape identifier** (es. *tape1*);
- il modello del nastro (che deve essere definito nel Tape Device Model Database);
- la descrizione del tipo dei file memorizzati nel nastro. Infatti, per rendere più agevoli le operazioni di accesso ai dati, si pone il vincolo che i file che si memorizzano in un nastro siano tutti dello stesso tipo;
- altre eventuali informazioni.

4.5 Definizione delle elaborazioni

La definizione delle elaborazioni che dovranno essere eseguite dal sistema si articola in tre fasi:

1. definizione di tipi di task;
2. definizione di production, utilizzando tipi di task precedentemente definiti;
3. submit di richieste computazionali (utilizzando production precedentemente definite), per il processing di dati definiti nel Data Database.

4.5.1 Definizione di tipi di task

Gli utenti del sistema DEMI devono essere in grado di definire tipi di task. Un task rappresenta un'attività elementare di elaborazione, che gestisce un particolare aspetto del problema complessivo. Un task può avere zero o più input, e zero o più output.

Deve essere messo a disposizione degli utenti un apposito tool, per definire i tipi di task attraverso rappresentazioni grafiche, come quella mostrata in figura 4.5.

Nella rappresentazione grafica l'ellisse rappresenta il tipo di task, mentre i rettangoli rappresentano i corrispondenti input/output (nell'esempio di figura il tipo di task *TaskTypeX* ha due input: *Input1* e *Input2*, e tre output: *Output1*, *Output2* e *Output3*).

Le definizioni dei tipi di task devono essere memorizzate in uno specifico database, chiamato **Task Database**. In questo database ogni tipo di task viene rappresentato dai seguenti campi:

- il nome del tipo di task (es. *TaskTypeX*);

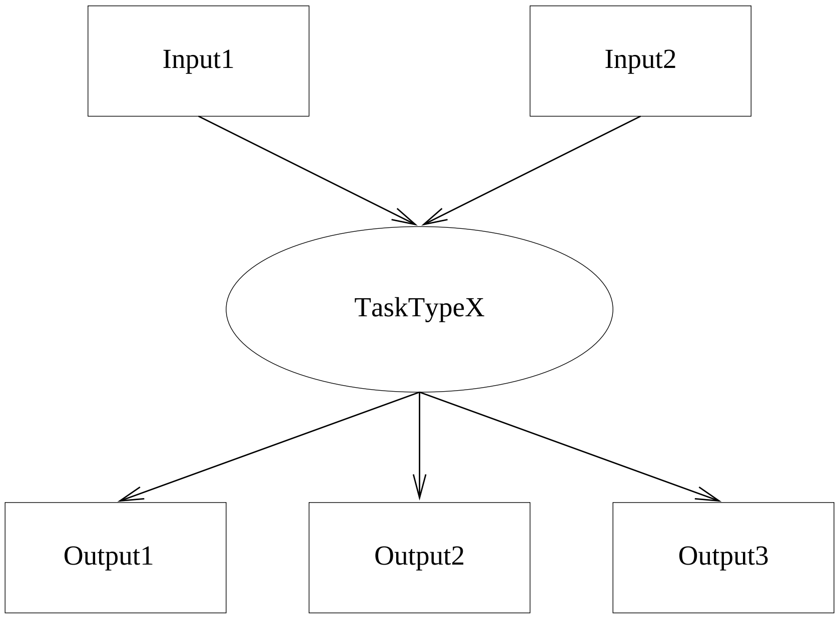


Figura 4.5: Esempio di rappresentazione grafica di un tipo di task

- la descrizione (es. *Task che gestisce l'attività X*);
- lo user identifier dell'utente che ha definito il tipo di task;
- gli input del tipo di task.

  È necessario specificare:

  - il numero di questi input;
  - il loro nome (es. *Input1*, *Input2*, ecc..);

- gli output del tipo di task.

  È necessario specificare:

  - il numero di questi output;
  - il loro nome (es. *Output1*, *Output2*, ecc..);

- la lista delle architetture di calcolatori per le quali il tipo di task considerato è stato implementato (le architetture da considerare sono quelle definite nell'Architecture Database).

  Per ogni elemento di questa lista è necessario definire:

- i file necessari per l'esecuzione del task (programma eseguibile, script che definiscono variabili logiche, ecc...); non devono invece essere specificati i file di input e di output;

- la linea di comando che deve essere specificata per eseguire il task.

  Nel programma utilizzato per l'esecuzione del task, si fa riferimento agli input e agli output attraverso dei "parametri formali" (ad esempio con il primo parametro si può fare riferimento all'input *Input1*, con il secondo parametro all'input *Input2*, ecc...). Quando si deve eseguire il task, utilizzando file di input e producendo file di output, i nomi di questi file di input/output costituiscono i "parametri attuali", che devono essere associati ai corrispondenti "parametri formali". I parametri attuali vengono specificati nella linea di comando usata per eseguire il task.

  Ad esempio, per il task di figura 4.5, nella linea di comando dovranno essere specificati almeno cinque parametri (è possibile che ne siano necessari anche altri): due per i due input *Input1* e *Input2*, e gli altri tre per gli output *Output1*, *Output2* e *Output3*. Ad esempio, se per l'input *Input1* deve essere usato il file *file1*, per l'input *Input2* si deve utilizzare il file *file2*, e per i tre output *Output1*, *Output2* e *Output3* si devono produrre rispettivamente i file con nome *out1file*, *out2file* e *out3file*, la linea di comando sarà del tipo:

  ```
  TaskTypeXExecFile    file1 file2 out1file \
                       out2file out3file
  ```

  È quindi necessario definire la corrispondenza tra i numeri ordinali dei parametri, e i corrispondenti nomi degli input-output;

- il parametro 'cost of the task';

  questo valore numerico rappresenta la stima del tempo necessario per l'esecuzione del task sull'architettura considerata. Non si tratta di un vero e proprio valore di tempo (ad esempio un valore pari a '10' non rappresenta 10 secondi, o 10 minuti): questo parametro è utilizzato per confrontare le diverse performance delle varie architetture nell'esecuzione del task considerato (un'architettura può essere più adatta rispetto ad un'altra nell'eseguire un certo task).

  Ovviamente possono esistere più calcolatori della stessa architettura con diverse prestazioni: questo aspetto viene considerato nel problema dello scheduling (vedi par. 4.7);

- altre eventuali informazioni.

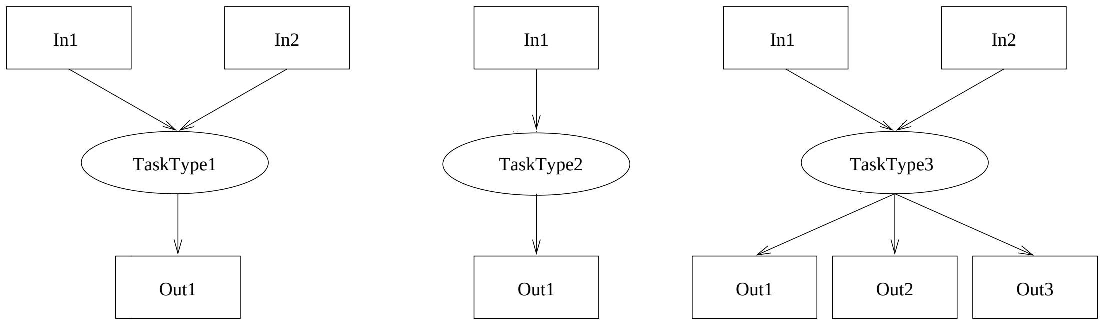


Figura 4.6: Esempi di tipi di task

Gli utenti del sistema DEMI devono poter essere in grado di definire nuovi tipi di task, di cancellare dal Task Database elementi precedentemente definiti, di modificare informazioni relative a tipi di task definiti nel database, di visualizzare gli elementi memorizzati nel Task Database, ecc...

4.5.2 Definizione di production

Utilizzando i tipi di task definiti nel Task Database, gli utenti del sistema DEMI possono definire production (produzioni): una production rappresenta un'elaborazione "complessa", composta da una serie di task in relazione tra loro. Quindi se un task gestisce un particolare aspetto del problema considerato, in una production si considera tale problema nella sua complessità.

Una production può essere composta da uno o più task, che possono anche essere (tutti o in parte) dello stesso tipo.

Anche in questo caso deve essere messo a disposizione degli utenti un apposito tool grafico per la definizione di production.

In figura 4.6 sono rappresentati tre tipi di task (*TaskType1*, *TaskType2* e *TaskType3*), che vengono utilizzati per la definizione della production di figura 4.7: il task *Task1* è di tipo *TaskType1*, il task *Task2* è di tipo *TaskType2*, il task *Task3* è ancora di tipo *TaskType1*, e il task *Task4* è di tipo *TaskType3*.

Nella definizione grafica di una production le ellissi rappresentano quindi i task che la compongono, mentre i rettangoli rappresentano i loro input/output, che vengono definiti come *data box* della produzione.

Nell'esempio di figura 4.7, i data box *Input1*, *Input2*, *Input3* e *Input4* sono input della produzione.

Come descritto nel paragrafo 4.5.3, le produzioni vengono poi usate per definire

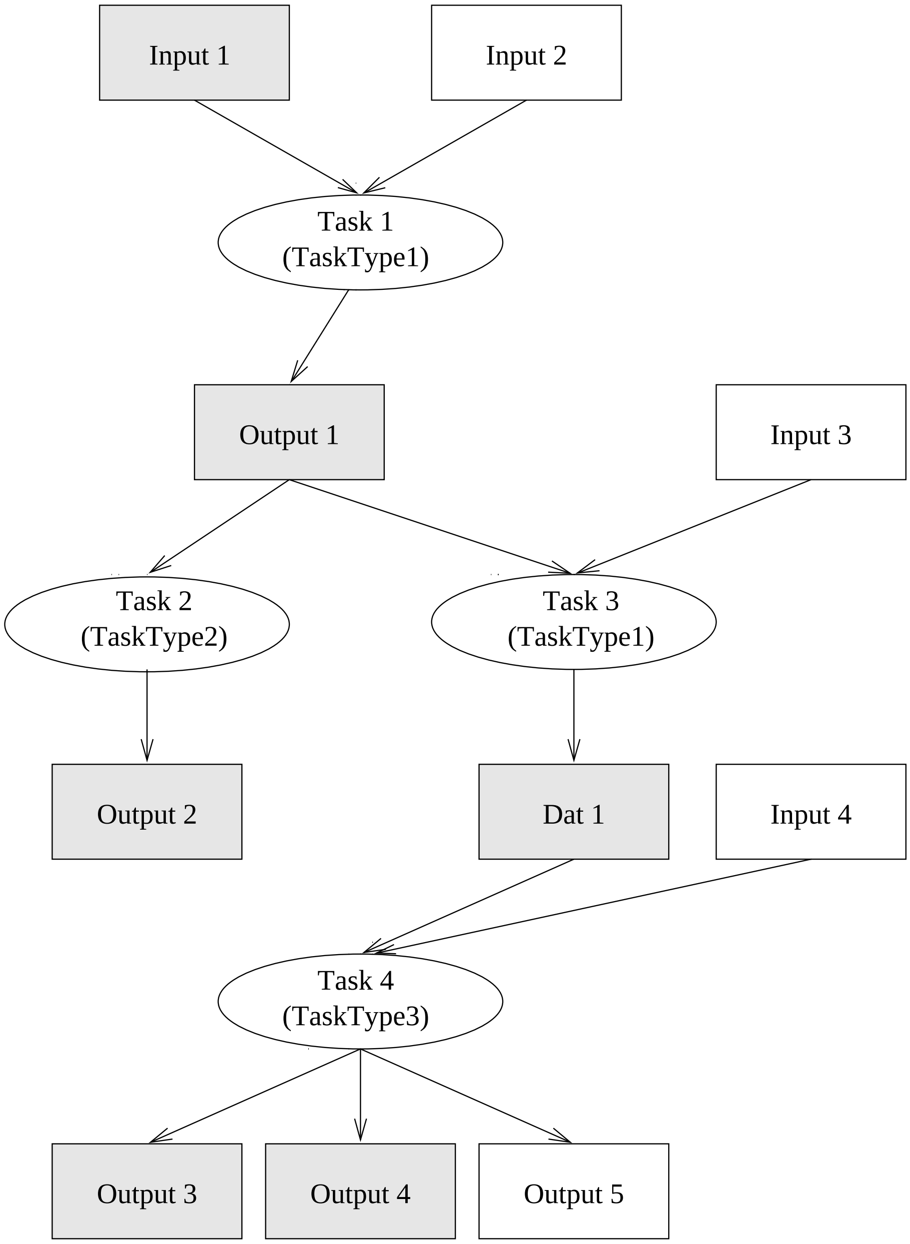


Figura 4.7: Esempio della rappresentazione grafica di una production

richieste computazionali, per elaborare specifici dati. Una produzione può essere utilizzata più volte, in momenti diversi, per elaborare dati diversi.

In base a come si "associano" i dati agli input di una produzione, questi vengono classificati in due categorie:

- input di tipo **stream**

  In una richiesta computazionale, ad un input di tipo stream di una produzione viene associata una lista di file, che devono essere elaborati uno alla volta. La collezione di questi file viene definita come *stream input set*, e ogni singolo file come *stream input file*;

- input di tipo **singleton**

  In una richiesta computazionale, ad un input di tipo singleton di una produzione viene associato un singolo file, chiamato *singleton input file*, unico elemento del *singleton input set*.

In modo analogo si possono classificare gli output della produzione (nell'esempio di figura 4.7 i data box *output1*, *output2*, *output3* e *output4*):

- output di tipo stream

  in una richiesta ad ognuno di essi è associato un insieme di file di output (*stream output set*), ognuno dei quali viene chiamato *stream output file*;

- output di tipo singleton

  in una richiesta ad ognuno di essi viene associato un singolo file (chiamato *singleton output file*).

Oltre ai data box input e output, esistono poi i data box a cui nelle richieste sono associati file "temporanei", che possono essere cancellati al termine dell'elaborazione, in quanto sono risultati intermedi che non si vogliono "salvare" (il data box *Dat1*, nell'esempio considerato).

Nella rappresentazione grafica proposta, un data box di tipo singleton è rappresentato da un rettangolo bianco, mentre uno di tipo stream è rappresentato attraverso un rettangolo "ombreggiato".

I dati di tipo stream corrispondono agli event data degli esperimenti HEP, mentre i dati di tipo singleton ai long term data.

Sempre con riferimento alla produzione di figura 4.7, si consideri una richiesta computazionale per cui all'input di tipo stream *Input1* viene associato lo stream input set costituito dalla seguente lista di stream file:

```
file1
file2
file3
file4
...
...
filen
```

mentre agli input di tipo singleton *Input2*, *Input3* e *Input4* vengono associati rispettivamente i file *singlefile2*, *singlefile3* e *singlefile4*, come rappresentato in figura 4.8.

Nell'elaborazione del task *Task1* si deve considerare il prodotto cartesiano *InpSet1* X *InpSet2*, dove *InpSet1* rappresenta lo stream input set associato all'input *Input1* della produzione, mentre *InpSet2* è il singleton input set (costituito dall'unico file *singlefile2*) associato all'input *Input2* della produzione: nella prima iterazione della computazione si deve elaborare la coppia di file *(file1, singlefile2)*, nella seconda iterazione si deve considerare la coppia *(file2, singlefile2)*, ecc... Nella prima iterazione verrà prodotto il file di output *out1file1* (per l'output di tipo stream *Output1*); nella seconda iterazione verrà generato il file *out1file2*, ecc... Per il task *Task3*, nella prima iterazione si considerano i file *out1file1* e *singlefile3* (rispettivamente associati ai data box *Output1* e *Input3*), producendo per il data box *Dat1* il file *dat1file1*; nella seconda iterazione si elaborano i file *out1file2* e *singlefile3*, generando il file *dat1file2*, ecc... Per quel che riguarda l'output *Output5*, di tipo singleton, il relativo file *outsinglefile* sarà il risultato di tutte le n iterazioni considerate.

Una production può avere:

- zero o un input di tipo stream;
- zero o più input di tipo singleton;
- zero o più output di tipo stream;
- zero o più output di tipo singleton.

Le definizioni delle production vengono memorizzate in un database, chiamato **Production Database**; ogni elemento di questo database viene descritto dai seguenti campi:

- il nome della production (es. *ProdY*);
- la descrizione (es. *Production che gestisce l'elaborazione Y*);

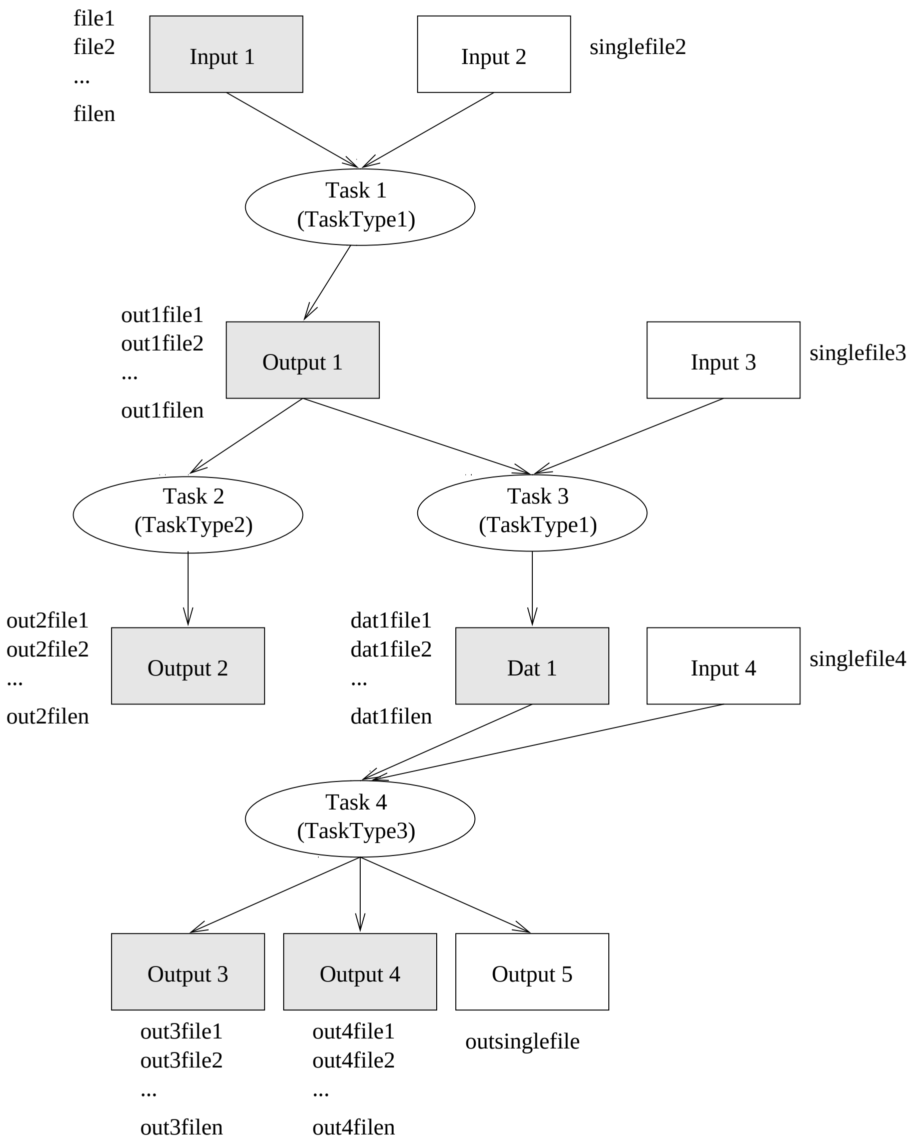


Figura 4.8: Esempio di utilizzo di una production in una computational request per l'elaborazione di dati

- lo user identifier dell'utente che ha definito la production;
- i task che compongono la production, specificandone i relativi tipi (questi tipi di task devono essere definiti nel Task Database);
- le relazioni che esistono tra i task componenti la production;
- i data box della production.

  È necessario specificare:

  - il loro numero;
  - il loro nome (es. *Input1*, *Input2*, *Dat1*, *Output1*, *Output2*, ecc...);
  - il "ruolo" di questi dati, che può essere:
    - dati di input (*InputX* nell'esempio di figura);
    - dati di output (*OutputX* nell'esempio di figura);
    - dati "temporanei" (*DatX* nell'esempio di figura);
      questi dati, una volta che non sono più necessari (sono stati usati come input da uno o più task) possono essere cancellati.
  - il loro tipo (stream o singleton);
- altre eventuali informazioni.

Oltre alla definizione di nuove production, gli utenti devono essere in grado di cancellare production precedentemente definite, di modificare informazioni relative a elementi presenti nel Production Database, di visualizzare gli elementi definiti nel database, ecc...

4.5.3 Submit di richieste computazionali

Come introdotto nel paragrafo precedente, ogni utente può sottomettere proprie richieste computazionali (computational request) al sistema DEMI, ovvero richieste di eseguire specifiche elaborazioni (rappresentate dalle production) su specifici dati, definiti nel Data Database.
Per sottomettere una computational request, l'utente deve definire:

- il nome della production da usare in questa elaborazione (es. *ProdY*).

  Questa production deve essere definita nel Production Database;

- gli input set (di tipo singleton e di tipo stream).

  Per ogni input di tipo singleton della corrispondente production è necessario specificare l'identificatore del file che deve essere usato (questo singolo file costituisce il corrispondente singleton input set).

  Per l'eventuale input di tipo stream (una production può avere al più un input di tipo stream) si deve definire il relativo stream input set, ovvero i file di dati che si devono utilizzare. Deve quindi essere specificata una lista del tipo:

  ```
  fileid1
  fileid2
  fileid3
  fileid4
  ...
  ...
  ```

  dove ogni riga rappresenta un file identifier (che deve essere definito nel Data Database).

  Se la production non ha input di tipo stream, è necessario specificare il numero di iterazioni, cioè il numero di volte in cui tale production deve essere eseguita;

- gli output (di tipo stream e di tipo singleton).

  Per ogni output della corrispondente produzione l'utente deve definire:

  – la descrizione del tipo dei file di output che saranno prodotti (es. *QQPS, 183 GeV, 97_1 MC*): questa informazione corrisponde al relativo campo del Global Data Database (vedi par. 4.4.1);

  – dove memorizzare i file che verranno generati; l'utente può scegliere di archiviare questi file di output:

    ∗ in una sua directory "privata";
    in questo caso dovrà essere specificato il pathname di questa directory;

    ∗ su nastri;
    in questo caso l'utente può definire il modello di nastri da utilizzare (modello che deve essere definito nel Tape Device Model Database). Questa opzione può comunque essere omessa, se non è richiesto l'utilizzo di uno specifico modello di nastri;

  * sui disk buffer gestiti dal sistema DEMI (vedi par. 4.3.2);

- la dimensione massima richiesta per un singolo file, associato ad ogni data box della production considerata.

  Questo parametro è necessario per stabilire quanto spazio deve essere allocato per memorizzare i file generati nei vari processing;

- un valore numerico, chiamato 'priority level', che definisce il livello di priorità per la computational request: un valore elevato di questo parametro indica la necessità di soddisfare prima possibile la richiesta;

- altre eventuali informazioni.

L'utente, quindi, non deve preoccuparsi di scegliere i file identifier per gli output che vengono prodotti. Questo viene fatto dal sistema che, una volta elaborata la richiesta, comunica all'utente gli identificatori dei file di output generati.

Tutti i dati relativi ad una richiesta vengono memorizzati in un database, chiamato **Request Database**.

Ad ogni computational request viene associato uno specifico identificatore, chiamato **REQID** (**REQuest IDentifier**), attraverso il quale l'utente può ad esempio verificare lo stato di elaborazione della richiesta, può ottenere la lista degli identificatori dei file di output prodotti fino a quel momento, ecc...

Come rappresentato in figura 4.9, il Task Database, il Production Database e il Request Database sono database "globali", di cui esiste un'unica istanza per tutto il sistema DEMI.
Nella figura sono anche schematizzate le fasi in cui si articola la definizione di una elaborazione:

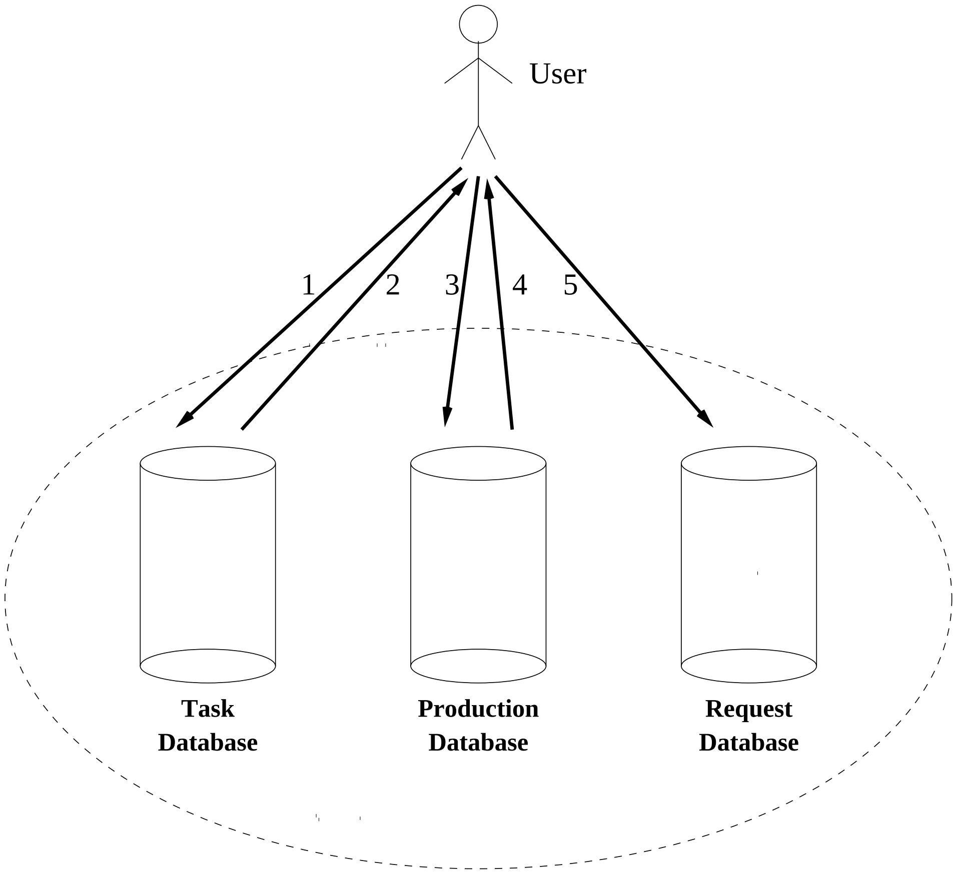
1. definizione di tipi di task;

2. selezione di tipi di task per la definizione di una production;

3. definizione di una production;

4. selezione di una production per la definizione di una computational request;

5. submit di una computational request.



1: Definizione di tipi di task

2: Selezione di tipi di task per la definizione di una production

3: Definizione di una production

4: Selezione di una production per la definizione di una computational request

5: Submit di una computational request

Figura 4.9: Definizione di una elaborazione

4.6 Elaborazione

La richiesta computazionale sottomessa dall'utente viene prima di tutto "scomposta" in **elementary request**: se la produzione corrispondente alla richiesta ha un input di tipo stream, ogni elementary request corrisponde ad un singolo stream file associato a questo input. Se invece la production non ha input di tipo stream, ogni elementary request corrisponde ad un'iterazione (il numero di iterazioni deve essere stato specificato dall'utente durante il submit della richiesta).

Si supponga, ad esempio, che la production corrispondente alla computational request considerata abbia due input di tipo singleton (*Input1* e *Input2*), e un input di tipo stream (*Input3*).

Si supponga che per il primo input di tipo singleton sia stato specificato il file:

```
singlefile1
```

mentre per il secondo input di tipo singleton sia stato specificato il file:

```
singlefile2
```

Si supponga infine che, per l'input di tipo stream, sia stato specificato come stream input set la seguente lista di file identifier:

```
fileid1
fileid2
fileid3
fileid4
...
...
fileidn
```

In questo caso la richiesta viene "scomposta" in n elementary request, a cui sono associati i seguenti file di input:

	INPUT1	INPUT2	INPUT3
el-req-1	singlefile1	singlefile2	fileid1
el-req-2	singlefile1	singlefile2	fileid2
el-req-3	singlefile1	singlefile2	fileid3
el-req-4	singlefile1	singlefile2	fileid4
...	...	...	...
...	...	...	...
el-req-n	singlefile1	singlefile2	fileidn

È importante sottolineare un aspetto, relativo agli input e agli output. Ad ogni input di tipo stream di una produzione viene associato un singolo file, per ogni elementary request. Per quel che riguarda gli input di tipo singleton, invece, ad ognuno di essi viene associato un file per ogni richiesta: questo file è quindi lo stesso per tutte le elementary request che "compongono" la richiesta. Quindi ogni stream input file fa riferimento ad una singola elementary request, mentre un singleton input file è associato alla computational request. Un discorso del tutto analogo si può fare per gli output: gli stream output file sono associati alle elementary request, mentre i singleton output file sono associati alle computational request.

Le elementary request vengono inserite nella cosiddetta **global queue of elementary requests**, una struttura dati in cui l'inserimento avviene in maniera ordinata rispetto al valore del parametro 'priority level': quando una nuova elementary request deve essere inserita nella global queue of elementary requests, essa viene posta dopo le elementary request con stessa o più elevata priorità precedentemente inserite, ma prima delle elementary request a più bassa priorità.

Le elementary request memorizzate nella global queue of elementary requests sono pronte per essere elaborate.

4.7 Scheduling

4.7.1 Scheduling globale

Le elementary request memorizzate nella global queue of elementary requests devono essere "assegnate" ai vari siti del sistema DEMI, per la loro elaborazione. Obiettivo principale che deve essere considerato in questo problema di mapping globale è la massimizzazione del *throughput*, cioè la "quantità" di elaborazione complessiva realizzata dal sistema in un certo intervallo di tempo.

Il mapping delle elementary request ai siti del sistema DEMI viene affidato ad un **global scheduler**, che deve analizzare queste elementary request, considerando prima quelle memorizzate in "testa" alla global queue of elementary requests (le elementary request con più elevato valore del parametro 'priority level'), e "associarle" ai siti più adatti per la loro elaborazione.

Le regole generali che il global scheduler deve considerare sono:

- una elementary request deve essere assegnata ad un sito in grado di elaborarla.

  Un'elementary request può essere elaborata in un sito:

  - se in quel sito tutti i task componenti la corrispondente production possono essere eseguiti (ci sono computer di tutte le architetture necessarie);
  - se in quel sito ci sono i necessari tape device in grado di leggere/scrivere da/su nastro i file di input/output;

- se la produzione associata ad una computational request ha uno più output di tipo singleton, tutte le elementary request componenti questa richiesta devono essere assegnate allo stesso sito (infatti per tutte queste elementary request si deve scrivere su stessi singleton output file).

Se non è possibile assegnare una elementary request a nessun sito (ad esempio perché un task, componente della corrispondente production, è stato implementato solo per una certa architettura, e in nessun sito vi sono calcolatori di questa architettura), l'elementary request non può essere elaborata: essa viene rimossa dalla global queue of elementary requests, e l'utente che ha sottomesso la corrispondente computational request viene avvisato riguardo il problema.

Lo scheduling globale è un problema molto complesso, in cui devono essere considerati vari aspetti, ad esempio:

- la potenza computazionale disponibile nei vari siti.

  Ovviamente in questo calcolo devono essere considerati solo i computer in grado di eseguire i task delle production considerate;

- i siti dove sono fisicamente presenti i dati necessari per l'elaborazione.

  Lo spostamento di dati tra i vari siti può infatti richiedere tempo e risorse non trascurabili: la dimensione di questi file può essere infatti molto elevata.

  Come già detto, per la gestione di questo problema deve essere implementato un *data mover*, che deve decidere se, quando e come (usando la Wide Area Network, usando spedizioni via corriere, ecc...) i dati devono essere spostati da un sito ad un altro;

Per la realizzazione del global scheduler è quindi necessario definire specifici criteri (quando può essere utilizzata la Wide Area Network per la copia dei dati, qual è la forma più conveniente di file transfer, ecc...), in cui sono richiesti appositi strumenti (ad esempio per il monitoring dello stato dei link della WAN in tempo reale). Come descritto nel capitolo 6, il global scheduler non è stato implementato nel primo prototipo realizzato.

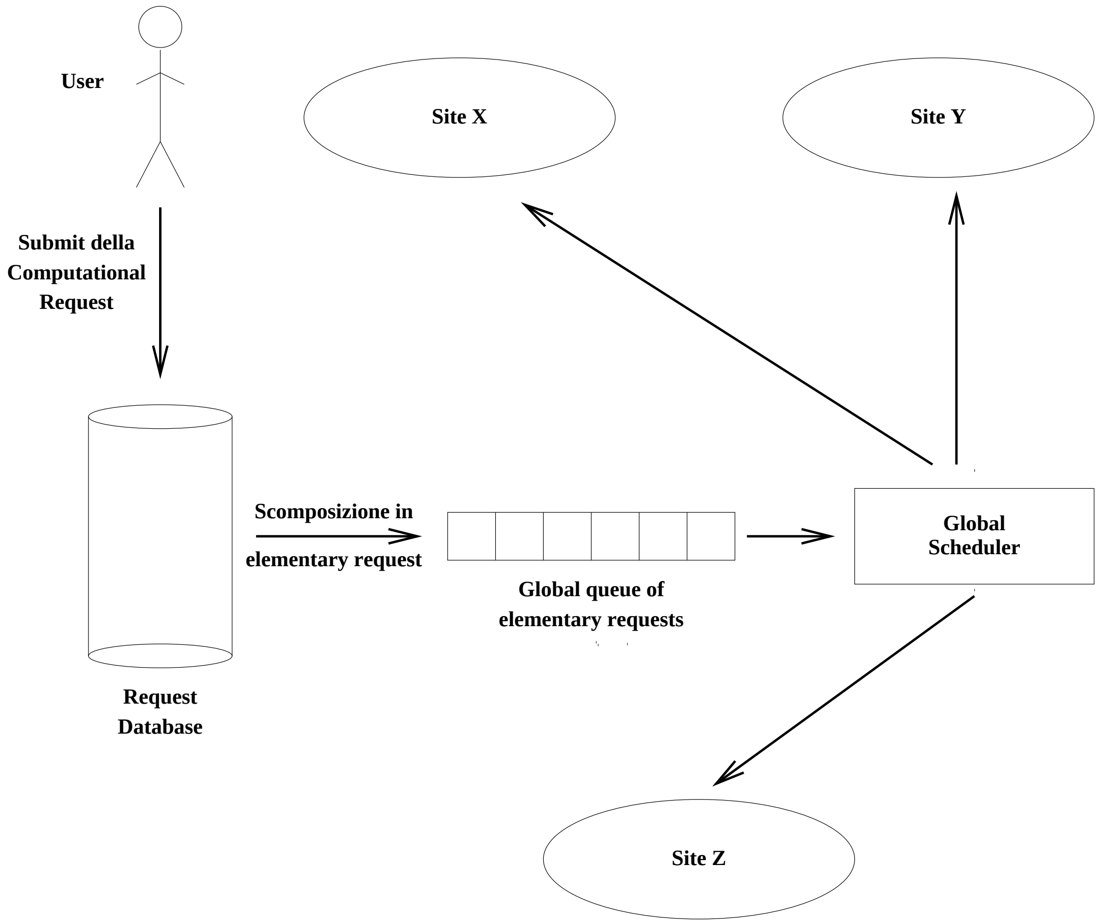


Figura 4.10: Sistema di scheduling globale

Una visione complessiva del sistema di scheduling globale è mostrata in figura 4.10: un utente, presente in un certo sito, sottomette una computational request: le relative informazioni vengono memorizzate nel Request Database. La computational request viene "scomposta" in elementary request, che vengono memorizzate nella global queue of elementary requests. Il global scheduler esamina queste elementary request, e per ognuna di esse sceglie il sito "migliore" per l'elaborazione.

4.7.2 Scheduling locale

Le elementary request assegnate ad un sito dal global scheduler, vengono memorizzate in una particolare coda, chiamata **local queue of elementary requests**. Esiste quindi una local queue of elementary requests per ogni sito del sistema DEMI.

La local queue of elementary requests ha la stessa struttura della global queue of elementary requests: le elementary request memorizzate in questa coda sono ordinate rispetto al valore del parametro 'priority level'. Quando una nuova elementary request deve essere inserita nella local queue of elementary requests, essa viene posta dopo le elementary request con stessa o più alta priorità precedentemente inserite, ma prima delle elementary request con più bassa priorità.

Le elementary request vengono analizzate, una alla volta, da un **local scheduler**, che considera per primi gli elementi memorizzati in "testa" alla local queue of elementary requests (le elementary request con più elevata priorità). Per ogni elementary request il local scheduler esamina quali sono i task che compongono la corrispondente production, e ognuno di essi viene "assegnato" ad un computer per l'elaborazione. Ovviamente l'esecuzione di un task può iniziare solo quando tutti i suoi input sono disponibili.

Si consideri ad esempio una elementary request che fa riferimento alla produzione rappresentata in figura 4.7. Per questa elementary request i file associati ai quattro input (*Input1*, *Input2*, *Input3* e *Input4*) siano rispettivamente *file1*, *singlefile2*, *singlefile3* e *singlefile4*, come rappresentato in figura 4.11 (i file *singlefile2*, *singlefile3* e *singlefile4* saranno usati anche per le altre elementary request della stessa computational request, trattandosi di singleton input file).

Il task *Task1* può iniziare subito l'esecuzione, considerando i file di input *file1* e *singlefile2*: verrà prodotto lo stream output file *out1file1*. Il task *Task2* può iniziare l'esecuzione solo dopo che il task *Task1* è terminato: come input viene considerato il file *out1file1*, e viene generato lo stream output file *out2file1*. Anche il task *Task3* può essere eseguito solo dopo il completamento dell'elaborazione del task *Task1*: usando i file di input *out1file1* e *singlefile3*, viene prodotto lo stream file *dat1file1*. Infine il task *Task4*, che utilizza come input i file *dat1file1* e *singlefile4*, può essere eseguito dopo l'esecuzione del task *Task3*. Dopo questa elaborazione, lo stream file temporaneo *dat1file1* può essere cancellato.

Obiettivo di questo problema di mapping locale è la massimizzazione del throughput, cioè la "velocità" complessiva con cui i dati di input vengono elaborati. Non sono invece posti vincoli riguardo la *latenza* (cioè il tempo necessario per l'elaborazione di un singolo input).
A tal scopo il local scheduler deve considerare vari parametri, quali:

- i requisiti di eterogenità.

  Un task può essere stato implementato solo per un certo numero di architetture; dovrà quindi essere scelto un calcolatore in grado di eseguire questa elaborazione;

- il tipo di architettura dei computer.

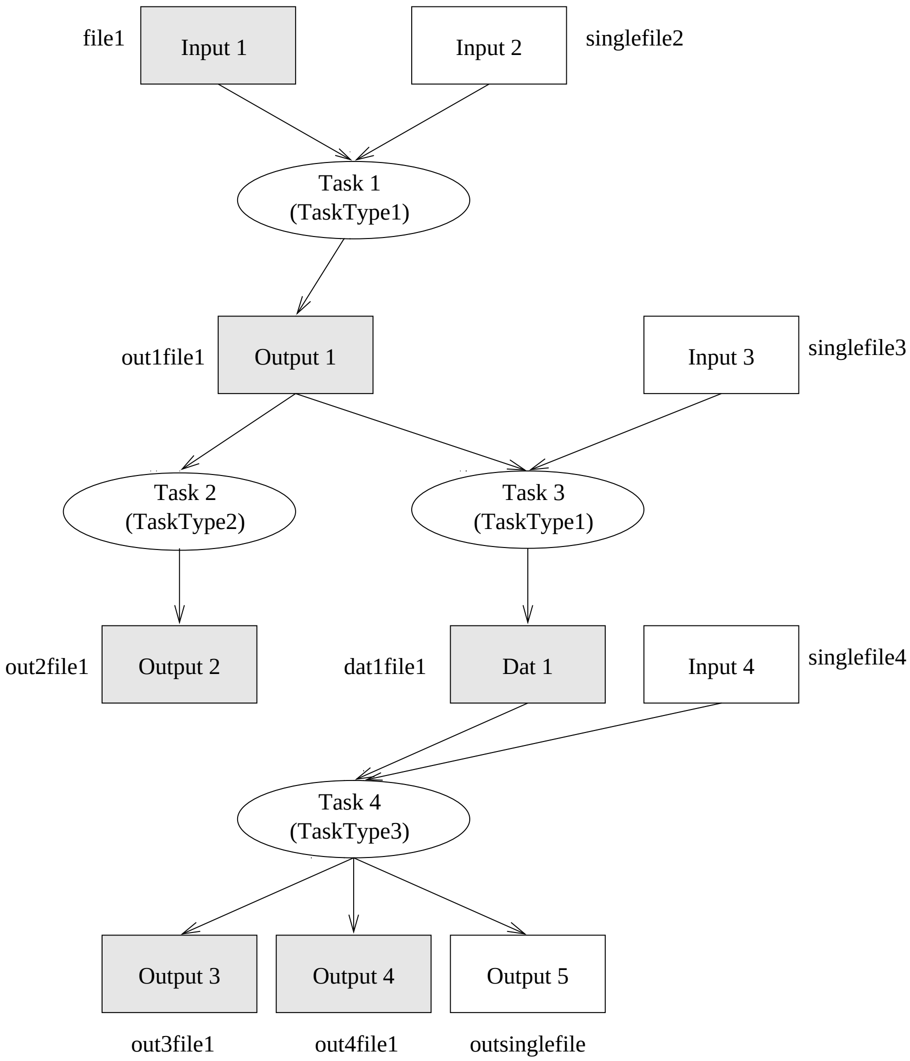


Figura 4.11: Elaborazione di una elementary request

Un task può essere stato implementato per più architetture, ma possono esserci architetture più adatte per l'elaborazione. Per questo scopo si considera l'attributo 'cost of the task' del Task Database;

- le performance dei computer.

  I vari computer possono avere diverse prestazioni: a tal fine il local scheduler deve considerare il parametro 'performances' del Local Computer Database;

- il *load* dei vari computer presenti nel sito, considerando anche quello non dovuto a job DEMI (job sottomessi da altri utenti).

Non è invece necessario considerare il problema della conversione dei dati, tra calcolatori di diversa architettura, e che quindi possono usare metodi diversi per la loro rappresentazione: si può infatti supporre che in tutti i computer si usino gli stessi formati standard per gestire i dati.

In generale un task non può essere in esecuzione su un computer che in quel momento è usato da un utente generico. Se un task DEMI inizia la sua esecuzione su un calcolatore che poi viene "reclamato" da un utente, l'esecuzione del task DEMI deve essere sospesa (o, eventualmente, il job può essere migrato su un altro calcolatore disponibile): l'elaborazione può essere ripresa solo quando l'utente ha terminato di usare tale computer. Questa "politica di scheduling" può comunque essere diversa tra i vari siti: ad esempio si potrebbe considerare un ambiente in cui le workstation devono essere utilizzate solo per il sistema DEMI, che quindi deve avere più elevata priorità rispetto ai processi interattivi degli utenti.

Quando un'elementary request è stata elaborata con successo, il sistema deve aggiornare il Data Database, per registare i nuovi stream output file che sono stati prodotti.

Quando è terminata l'elaborazione di una richiesta computazionale (cioè tutte le "sue" elementary request sono state elaborate), il Data Database deve essere aggiornato con i relativi singleton output file. L'utente che aveva sottomesso tale richiesta deve essere avvertito, e gli deve essere comunicata la lista degli identificatori dei file di output generati.

Una visione del sistema di scheduling locale è presentata in figura 4.12: le elementary request assegnate ad un certo sito dal global scheduler vengono memorizzate nella local queue of elementary requests. Il local scheduler esamina queste elementary request, e sceglie i computer più adatti dove eseguire i corrispondenti task.

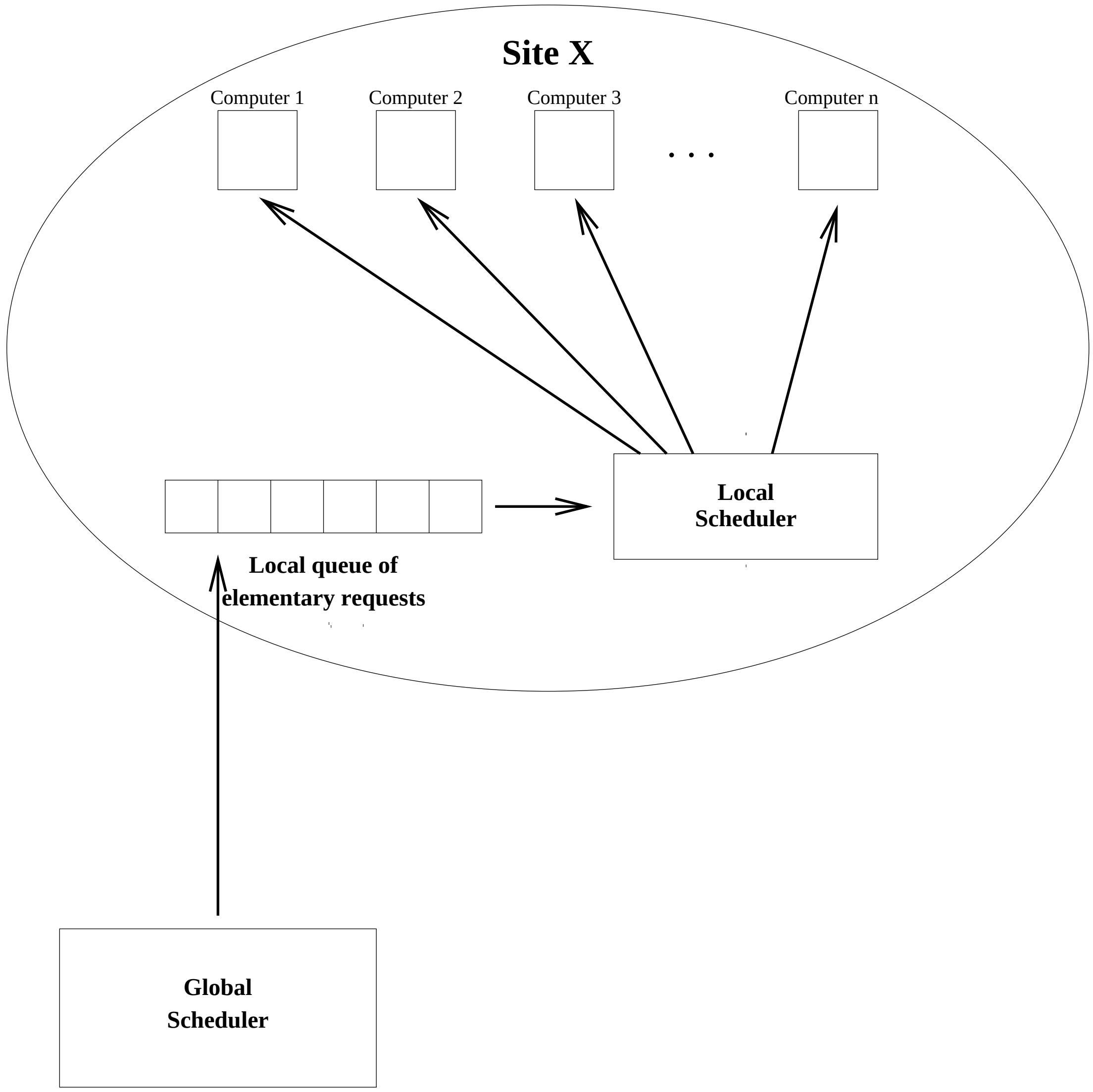


Figura 4.12: Sistema di scheduling locale

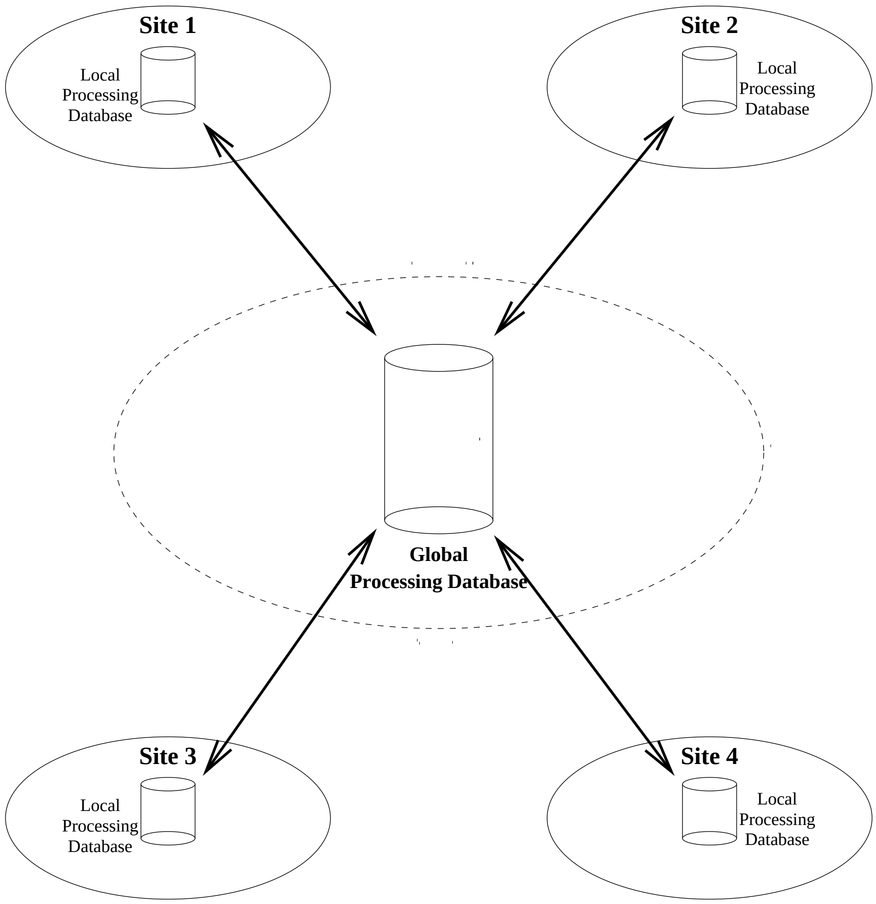


Figura 4.13: Processing Database

4.8 Processing Database

Per tener traccia di tutte le elaborazioni eseguite nei diversi siti del sistema DEMI, deve essere utilizzato uno specifico database, chiamato **Processing Database**.

La struttura del Processing Database è mostrata in figura 4.13: esso consiste quindi di un **Global Processing Database** "centrale" e, in ogni sito, di un **Local Processing Database**.

4.8.1 Global Processing Database

Ogni elemento del Global Processing Database rappresenta un **elementary processing**, ovvero l'elaborazione di una elementary request.

Un elementary processing è rappresentato nel Global Processing Database dai seguenti campi:

- l'identificatore della elementary request (**elementary request identifier**);
- il REQID (REQuest IDentifier) della corrispondente richiesta computazionale;
- il sito dove il processing è stato eseguito;
- lo stato dell'elaborazione (in esecuzione, elaborazione completata con successo, processing fallito, ecc...);
- altre eventuali informazioni.

4.8.2 Local Processing Database

In ogni sito del sistema DEMI è presente un Local Processing Database. Ogni elemento di questo database rappresenta l'elaborazione di una elementary request eseguita in quel sito, ed è descritto dai seguenti campi:

- l'elementary request identifier;
- per ogni elaborazione di ogni task componente la corrispondente production, deve essere specificato:
  - il nome del task;
  - il computer dove il processing è stato eseguito;
  - la data e l'ora in cui è iniziata l'elaborazione;
  - la data e l'ora in cui il processing è terminato;
  - lo stato dell'elaborazione (in esecuzione, elaborazione completata con successo, processing fallito, ecc...).

  È importante notare come, per uno stesso singolo task relativo ad una elementary request, potrebbero essere effettuate più elaborazioni: infatti un processing potrebbe non essere stato completato con successo (ad esempio per un crash del computer), e quindi eseguito ancora;
- altre eventuali informazioni.

Si supponga, ad esempio, che un'elementary request sia stata elaborata nel sito *SiteX*. Nel Global Processing Database deve essere presente un elemento relativo a questo elementary processing, in cui è appunto indicato che l'elaborazione è avvenuta nel sito *SiteX*. Nel Local Processing Database, implementato nel sito *SiteX*, dovrà essere presente una entry relativa a questo processing, a cui saranno associate tutte le corrispondenti informazioni.

Quindi nel Global Processing Database vi è un elemento per ogni elementary processing gestito dal sistema DEMI, mentre in ogni Local Processing Database c'è un elemento per ogni elementary processing effettuato nel corrispondente sito.

Gli utenti del sistema DEMI devono poter essere in grado di effettuare ricerche di elementi del Processing Database, utilizzando in questa query uno o più campi. Il sistema deve aggiornare automaticamente i vari attributi del Processing Database dopo ogni processing: non deve essere il singolo utente ad eseguire manualmente queste operazioni.

4.9 Logbook

In ogni sito del sistema DEMI viene utilizzato un **Local Logbook**, per tener traccia di tutti gli eventi significativi e di tutte le condizioni di errore avvenuti nel sito.
Ogni elemento di questo database è rappresentato dai seguenti campi:

- **categoria**;

  questo campo può essere uguale a:

  - 'info';
    definisce un evento "normale" occorso nel sistema;
  - 'error';
    definisce una condizione di errore;

- **tipo**;

  una sorta di "titolo", che descrive brevemente l'evento (es. *NewComputer*);

- **descrizione**;

  la descrizione "completa" dell'evento/errore (es. *Il local administrator X ha aggiunto al sito il computer Y*);

- **data**;

  la data in cui si è verificato l'evento;

- **ora**;

  l'ora in cui si è verificato l'evento.

Oltre ai Local Logbook, deve essere presente un **Global Logbook** "centrale", usato per memorizzare le informazioni relative agli eventi significativi e agli errori avvenuti nel sistema a livello globale. Il Global Logbook ha esattamente la stessa struttura del Local Logbook.

Il Global Logbook deve essere gestito unicamente dal global administrator, mentre i Local Logbook sono gestiti solamente dai corrispondenti local administrator. Gli amministratori possono effettuare su questi logbook operazioni di ricerca, di cancellazione di elementi, ecc...