Consiglio Nazionale delle Ricerche

SQL/DS:

Introduzione all'utilizzo

del sistema SQL/DS

2ª PARTE

R. Bartoli - A. Ceccarelli - O. Signore

203
CNUCE

A cura di : R. Bartoli
A. Ceccarelli
O. Signore

SQL/DS:
INTRODUZIONE ALL'UTILIZZO DEL SISTEMA SQL/DS
II PARTE

R. Bartoli - A. Ceccarelli - O. Signore

INDICE

1.0 Introduzione 1

2.0 Nuove possibilita' offerte dai comandi SQL e ISQL . . . 3
2.1 Scelta di dati da due o piu' tabelle 3
2.2 Uso di label nelle interrogazioni 6
2.3 Join di una tabella con se stessa 6
2.4 Informazioni riepilogative per gruppi 7
2.5 La clausola Having 8
2.6 Sottointerrogazioni successive per costruire condizioni di ricerca 9
2.7 Sottointerrogazioni correlate per costruire condizioni di ricerca 12
2.8 Uso delle viste per semplificare le interrogazioni . . . 13
2.9 Denominazione di colonne in una view 15

3.0 Struttura logica di un data base 17

4.0 Gestione delle proprie tabelle 21
4.1 Creazione di proprie tabelle 21
4.2 Dove vengono memorizzate le tabelle 22
4.3 Identificazione del contenuto minimo di una tabella . . 22
4.4 Eliminazione di una tabella 23
4.5 Copiatura di dati da altre tabelle 23
4.6 Aggiunta di una colonna a una tabella 23
4.7 Uso di proprie tabelle in comune con altri utenti . . . 23
4.7.1 Come concedere ad altri di selezionare le vostre tabelle 23
4.7.2 Concessione di altri privilegi sulle vostre tabelle 24
4.7.3 Concessione di tutti i privilegi ad un utente . . . 24
4.7.4 Concessione di privilegi a piu' utenti 24
4.7.5 Revoca dei privilegi concessi 24
4.8 Accesso alle tabelle che appartengono ad altri utenti 25
4.9 Indicizzazione di una tabella 25

5.0 Creazione ed uso delle routines 27
5.1 Come stabilire dove sono memorizzate le routines. . . . 27
5.2 Memorizzazione di una routine 28
5.3 Esecuzione di una routine 28
5.4 Uso di comandi Select in una routine 28
5.5 Le routines Profile 29

6.0 APPENDICE 31
6.1 Tabella Inventory 31
6.2 Tabella Suppliers 31
6.3 Tabella Quotations 32

Indice

ELENCO DELLE FIGURE

Figura 1. Join di due tabelle 4
Figura 2. I step di una operazione di Join 5
Figura 3. II step della operazione di Join 6
Figura 4. Risultato finale della operazione di Join 6
Figura 5. Uso di Label 7
Figura 6. Informazioni per gruppi 8
Figura 7. La clausola Having 9
Figura 8. Esempio di sottointerrogazione 9
Figura 9. Esempio di lista in una sottointerrogazione . . . 11
Figura 10. Sottointerrogazioni correlate 13
Figura 11. Interrogazione di una view 14
Figura 12. Interrogazione di una seconda view 15
Figura 13. Interrogazione di una terza view 15
Figura 14. Tabella ed indici memorizzati in un DBSPACE . . . 18
Figura 15. Rappresentazione logica di un Data Base 19
Figura 16. Routine Profile n.1 28
Figura 17. Routine Profile n.2 29

1.0 INTRODUZIONE

In questa parte del manuale sono riportati strumenti di lavoro per coloro che dovendo accingersi a questa nuova attivita' sono gia' in possesso dei fondamenti del linguaggio SQL.
In particolare oltre ad informazioni sulla logica di un Data Base sono fornite indicazioni sul come creare, gestire ed eventualmente mettere a disposizione di altri utenti i dati memorizzati nella propria banca dati sotto forma di tabelle.
L'uso del manuale comporta percio' aver utilizzato la I parte come supporto per un approccio alle conoscenze del sistema SQL/DS.

2 SQL/DS: Introduzione all'utilizzo del sistema SQL/DS II PARTE

2.0 NUOVE POSSIBILITA' OFFERTE DAI COMANDI SQL E ISQL

2.1 Scelta di dati da due o piu' tabelle

Il sistema SQL/DS consente di effettuare, in modo interattivo ad un numero variabile di utenti, una varieta' di operazioni sulle tabelle. Le operazioni base sono:

- creazione o cancellazione di tabelle
- operazioni di query
- aggiornamento, inserimento o cancellazione di dati
- aggiunta di nuove colonne a una tabella
- copiare dati da una tabella in un'altra
- eseguire operazioni di utility sulle tabelle (loading dei dati, riorganizzazione dei dati,...)

Una operazione particolarmente interessante, disponibile nel modello relazionale, e' chiamata "Joining". La funzione "Join" di SQL/DS permette la combinazione di due o piu' tabelle (max 16) basata sui valori contenuti nelle tabelle stesse per una operazione di query. Concettualmente SQL/DS esegue tutte le possibili combinazioni di righe delle tabelle indicate nella clausola From e per ogni combinazione verifica la "Join condition" specificata nella clausola Where. Se una riga di una delle tabelle non soddisfa la condizione di Join quella riga non appare nel risultato di Join.
Se non viene specificata una "Join condition", SQL/DS restituisce tutte le possibili combinazioni di righe delle tabelle specificate nella clausola From.

Facciamo ancora riferimento alle tabelle riportate in appendice e vediamo un esempio. Vogliamo conoscere i numeri di pezzo, le descrizioni ed i prezzi del materiale approvvigionato dal fornitore numero 54.
Le informazioni relative sono contenute nelle due tabelle Inventory e Quotations. Il comando da emettere e' il seguente:

```
select inventory.partno,description,price -
   from  quotations,inventory -
   where inventory.partno=quotations.partno -
   and  suppno=54
```

Se piu' di una tabella contiene un uguale nome di colonna, per identificare esattamente quale colonna si vuole referenziare e' necessario specificare il nome della tabella come prefisso. E'

ovvio come un nome unico di colonna non richiede il nome della tabella a cui appartiene.
Il risultato sara':

PARTNO	DESCRIPTION	PRICE
209	CAM	18.00
221	BOLT	0.10
231	NUT	0.04
241	WASHER	0.02

Figura 1. Join di due tabelle

Analizziamo come lavora il sistema per seguire una operazione di Join. Ogni valore della colonna Partno della tabella Inventory e' confrontato con ogni valore di Partno nella tabella Quotations. Quando entrambi i campi delle tabelle risultano uguali viene formata una nuova riga che contiene i campi combinati delle righe "matching". In questo caso il risultato sara':

SUPPNO	PARTNO	PRICE	DELIVERY_TIME	QONORDER	DESCRIPTION	QONHAND
64	207	29.00	14	20	GEAR	75
54	209	18.00	21	0	CAM	50
64	209	19.50	7	7	CAM	50
51	221	0.30	10	50	BOLT	650
54	221	0.10	30	150	BOLT	650
61	221	0.20	21	0	BOLT	650
53	222	0.25	15	0	BOLT	1250
61	222	0.20	21	200	BOLT	1250
51	231	0.10	10	0	NUT	700
54	231	0.04	30	200	NUT	700
53	232	0.10	15	200	NUT	1100
53	241	0.08	15	0	WASHER	6000
54	241	0.02	30	200	WASHER	6000
61	241	0.05	21	0	WASHER	6000
57	285	21.00	14	0	WHEEL	350
57	295	8.50	21	24	BELT	85
70	301	0.20	15	100	FUSE	300
71	301	0.25	10	0	FUSE	300
72	301	0.15	20	0	FUSE	300
70	302	0.21	16	0	FUSE	367
71	302	0.26	11	200	FUSE	367
72	302	0.16	21	0	FUSE	367
70	310	40.90	30	0	PIN	150
71	310	55.00	20	0	PIN	150
72	310	43.10	23	300	PIN	150
70	311	41.00	31	100	PIN	292
71	311	56.00	21	0	PIN	292
72	311	48.00	24	0	PIN	292
70	315	1.00	8	0	SWITCH	421
71	315	1.10	3	65	SWITCH	421
72	315	1.49	15	0	SWITCH	421
70	316	1.01	7	0	SWITCH	342
71	316	1.11	16	0	SWITCH	342
72	316	1.50	14	850	SWITCH	342
70	320	3.00	18	0	REALY	0
71	320	3.20	18	0	RELAY	0
72	320	3.50	11	0	RELAY	0
70	321	3.10	17	0	RELAY	0
71	321	3.19	19	0	RELAY	0
72	321	3.51	12	0	RELAY	0
70	322	3.15	16	0	RELAY	0
71	322	3.21	20	0	RELAY	0
72	322	3.49	13	0	RELAY	0
72	323	3.52	14	0	RELAY	0

Figura 2. I step di una operazione di Join

Successivamente, allorche' viene eseguita la seconda parte della clausola Where si ha:

SUPPNO	PARTNO	PRICE	DELIVERY_TIME	QONORDER	DESCRIPTION	QONHAND
54	209	18.00	21	0	CAM	50
54	221	0.10	30	150	BOLT	650
54	231	0.04	30	200	NUT	700
54	241	0.02	30	200	WASHER	6000

Figura 3. II step della operazione di Join

Il risultato e' ridotto e solo quelle righe con Suppno=54 vengono mantenute e l'intera "search condition" e' ora soddisfatta. Infine le colonne che SQL/DS restituisce come risultato della query sono quelle specificate nella "select-list" quindi avremo:

PARTNO	DESCRIPTION	PRICE
209	CAM	18.00
221	BOLT	0.10
231	NUT	0.04
241	WASHER	0.02

Figura 4. Risultato finale della operazione di Join

2.2 Uso di label nelle interrogazioni

L'interrogazione precedente puo' essere scritta in modo piu' semplice per mezzo di Label. E' possibile associare ad ogni tabella, usata nella operazione di Join, una Label successivamente utilizzata nella specifica degli attributi della interrogazione. Nel nostro caso si ha:

```
select  i.partno,description,price -
   from inventory i,quotations q -
   where  i.partno=q.partno -
   and   suppno=54   -
   order by i.partno
```

Nell'esempio la lettera "i" viene usata per fare riferimento alla tabella Inventory e la lettera "q" per fare riferimento alla tabella Quotations. La Label deve iniziare con una lettera e puo' contenere al max 18 caratteri. Il nome della tabella e della Label nella clausola From devono esser separate da uno spazio.

2.3 Join di una tabella con se stessa

Tramite la Label definita precedentemente, e' possibile fare riferimento ad una singola tabella come se si trattasse di due o piu' tabelle diverse. Per dire a SQL/DS che il Join consiste di combinazioni di righe della stessa tabella e' sufficiente ripetere il nome della tabella due o piu' volte nella clausola From.

Dovendo pero' essere un nome unico sara' necessario associare a questi una Label.

Volendo conoscere il prezzo combinato dei pezzi di codice 231 e 221 per ciascun fornitore che vende entrambi i pezzi dare il comando:

```
select  qa.suppno,qa.price+qb.price -
  from  quotations qa,quotations qb -
  where  qa.suppno=qb.suppno -
  and    qa.partno=231   and  qb.partno=221 -
  order by 2
```

fornisce il risultato:

SUPPNO	EXPRESSION 1
54	0.14
51	0.40

Figura 5. Uso di Label

Cioe' la tabella Quotations e' trattata come se fosse costituita da due diverse tabelle (Quotations A e Quotations B). In Quotations A vengono cercate le righe con il codice Partno=231 e in Quotations B le righe con il codice Partno=221. Le informazioni dalle due tabelle, con lo stesso numero di fornitore qa.suppno=qb.suppno, sono infine riunite in un unico report.

2.4 Informazioni riepilogative per gruppi

Supponiamo voler conoscere il prezzo medio di ciascun codice parte nella tabella Quotations. Un sistema e' quello di fare tanti comandi di Select specificando ogni volta nella clausola Where il numero di codice. L'altro metodo e' fornire una sola Select aggiungendo la clausola Group By. Nel nostro caso avremo:

```
select partno,avg(price) -
  from quotations   -
  group by partno  -
  order by partno
```

In questo modo SQL/DS raggruppa tutte le righe che contengono lo stesso numero di pezzo e calcola il prezzo medio che costituira' una riga della interrogazione. Successivamente seleziona, raggruppa e calcola il prezzo medio di un'altra parte e produce un'altra riga e cosi' via fino a che non sono stati selezionati tutti i numeri di parte.
Il risultato del comando e':

PARTNO	AVG(PRICE)
207	29.000000000000
209	18.750000000000
221	0.200000000000
222	0.225000000000
231	0.070000000000
232	0.100000000000
241	0.050000000000
285	21.000000000000
295	8.500000000000
301	0.200000000000
302	0.210000000000
310	46.333333333333
311	48.333333333333
315	1.196666666666
316	1.206666666666
320	3.233333333333
321	3.266666666666
322	3.283333333333
323	3.520000000000

Figura 6. Informazioni per gruppi

E' opportuno ricordare che la clausola Group By nella specifica segue sempre la clausola From ed identifica quale colonna si deve usare per raggruppare i risultati. Inoltre tutte le colonne incluse nella clausola Select devono avere una funzione incorporata associata o devono comparire nella clausola Group By.

2.5 La clausola Having

Specificando la clausola Having dopo la clausola Group By viene applicata una condizione di ricerca ai gruppi facendo in modo che il sistema fornisca il risultato solo per quei gruppi che soddisfano la condizione. Cioe' la clausola Having puo' essere pensata come la clausola Where applicata per gruppi.
Per esempio, per trovare il prezzo medio di quelle parti che nella tabella Quotations compaiono su piu' di una riga il comando e':

```
select partno,avg(price) -
  from quotations  -
  group by partno  -
  having count(*) > 1 -
  order by partno
```

Cio' da luogo a:

PARTNO	AVG(PRICE)
209	18.750000000000
221	0.200000000000
222	0.225000000000
231	0.070000000000
241	0.050000000000
301	0.200000000000
302	0.210000000000
310	46.333333333333
311	48.333333333333
315	1.196666666666
316	1.206666666666
320	3.233333333333
321	3.266666666666
322	3.283333333333

Figura 7. La clausola Having

2.6 Sottointerrogazioni successive per costruire condizioni di ricerca

Nelle operazioni di query viste fino ad ora, la clausola Where conteneva una o piu' condizioni di ricerca che SQL/DS usava per trovare le righe relative alle espressioni specificate nella Select-list. Il sistema SQL/DS permette di specificare una query chiamata Subquery che fa riferimento ad un valore o set di valori calcolati con un'altra query.
Supponiamo di voler trovare i numeri parte 221 in cui il prezzo e' maggiore di 2 volte il minimo prezzo per quella parte. Il problema comporta manifestamente l'esecuzione di due query: la prima per trovare il minimo prezzo per il codice di parte 221 e la seconda per trovare il codice di parte in cui il prezzo e' maggiore del risultato della prima query. Cioe':

```
select suppno,price -
  from quotations  -
  where partno=221  -
  and  price >   -
             (select 2*min(price) -
                from quotations -
                where  partno=221)
```

SUPPNO	PRICE
51	0.30

Figura 8. Esempio di sottointerrogazione

E' importante osservare che il risultato della subquery e' sostituito direttamente nel predicato di livello superiore in cui la subquery appare.
In ogni caso una subquery deve avere solo una singola colonna o espressione nella sua Select-list e non deve avere una clausola Order By.
Una subquery puo' ritornare alla query principale 3 diversi valori:

1. Subquery che ritorna un singolo valore: in questo caso il valore puo' essere usato sul lato destro di un predicato nella clausola Where o nella clausola Having (Subquery non sono permesse nei predicati Between).

2. Subquery che ritorna un valore nullo: in questo caso il predicato di livello superiore contenente la subquery valuta come "unknown" il valore vero. Vediamo allora come SQL/DS gestisce il valore "unknown". Consideriamo questo predicato:

```
Qonhand + Qonorder < 100
------------------   ----
  espressione1       espressione2
```

allora se Qonhand o Qonorder e' nullo, SQL/DS considera l'espressione 1 come valore nullo e il valore vero del predicato e' considerato unknown. Se infine questo predicato viene combinato con altri predicati usando gli operatori And, Or e Not, SQL/DS processa il valore unknown in base alla tabella di verita' seguente:

```
AND | T  F  ?        OR | T  F  ?        NOT |
----------------     -----------------     --------
 T  | T  F  ?         T  | T  T  T          T  | F
 F  | F  F  F         F  | T  F  ?          F  | T
 ?  | ?  F  ?         ?  | T  ?  ?          ?  | ?
```

dove "?" rappresenta il valore non determinato.

Supponiamo adesso che SQL/DS stia cercando in una tabella le righe che soddisfano la condizione:

```
price + 5.25 < 20.0 and suppno=51
```

allorche' SQL incontra una riga in cui il campo Price e' nullo ma Suppno=51, l'espressione Price+5.25 viene considerata nulla causando per questo il valore vero del primo predicato unknown. Il campo Suppno per quella riga e' 51 cosi' il secondo predicato e' vero. Facendo riferimento alla tabella AND abbiamo che i valori "True e unknown" causano la "search condition" nella query essere unknown; percio' la riga non soddisfa la condizione di ricerca.

3. Nel caso in cui la Subquery ritorna una lista di valori; nella query di livello superiore viene usato l'operatore In o Not In per selezionare le righe contenenti almeno uno dei valori della lista.

Vediamo due esempi. Supponiamo di voler cercare nella tabella dell'inventario le parti approvvigionate dal fornitore 53.
Per far cio' dovremo prima consultare la tabella Quotations e vedere che il fornitore 53 fornisce le parti 222, 232 e 241.
Potremo quindi scrivere:

```
select   *  -
  from inventory  -
  where partno in (select partno -
                     from quotations  -
                     where suppno=53)
```

In questo modo il secondo comando Select (subquery) fornisce la lista dei numeri parte della tabella Quotations dove il numero del fornitore e' 53. La lista cosi' fornita viene usata dal primo comando Select per selezionare le righe desiderate dalla tabella Inventory. Il risultato e':

PARTNO	DESCRIPTION	QONHAND
222	BOLT	1250
232	NUT	1100
241	WASHER	6000

Figura 9. Esempio di lista in una sottointerrogazione

Il sistema permette di fornire fino a 16 interrogazioni in cascata usate con uno qualsiasi degli operatori (=, ¬, ...) per collegarle alla interrogazione di livello piu' elevato.
Vediamo un esempio in cui sono riportate 3 query:

```
select * -
 from inventory -
 where partno in (select partno -
                    from quotations  -
                    where suppno=(select suppno -
                                    from suppliers -
                                    where name='atlantis co.')) -
 order by partno
```

In questo modo il sistema trova il numero di fornitore per Atlantis Co. nella tabella Suppliers quindi, nella tabella Quotations i numeri parte per quel fornitore ed infine seleziona dalla tabella Inventory le righe per i codici parte trovati.

2.7 Sottointerrogazioni correlate per costruire condizioni di ricerca

Negli esempi precedenti abbiamo visto che la subquery e' valutata una sola volta ed il risultato e' sostituito nel predicato della query di livello superiore. Riprendiamo un esempio di interrogazione gia' visto:

```
select  suppno,price -
   from  quotations  -
   where  partno=221 -
   and  price  >  -
                (select  2*min(price) -
                    from quotations  -
                    where partno=221 )
```

in questo caso ricordiamo vengono cercate le quotazioni per il numero parte 221 in cui il prezzo e' maggiore di 2 volte il minimo prezzo per il numero parte. Consideriamo ora il problema:

trovare le quotazioni per ogni numero parte in cui il prezzo e' piu' di due volte il minimo prezzo per quel numero parte.

Questo significa che per **ogni numero parte** nella tabella deve essere calcolato il prezzo minimo.
In questo caso sara' necessario riferire una condizione di ricerca in una sottointerrogazione ad un valore di ciascuna riga della tabella indicata nella interrogazione principale. Occorre cioe' correlare la selezione nell'interrogazione principale con la condizione di ricerca nella sottointerrogazione.

Il concetto di correlazione nel sistema SQL/DS permette di scrivere una subquery che e' eseguita ripetutamente una volta per ogni riga della tabella identificata nella query di livello superiore mentre nella successione di interrogazioni viste nel paragrafo precedente il sistema valuta la sottointerrogazione una volta (per una particolare parte) e quindi usa il risultato nella interrogazione principale.

La nostra query diventa:

```
select  suppno,partno,price  -
  from  quotations  x-
  where  price  >  (select 2*min(price) -
                      from quotations  -
                      where  partno=x.partno)
```

In definitiva la clausola **Where partno=x.partno** dice al sistema di calcolare la subquery una volta per ogni Partno nella tabella della query di livello superiore.

SUPPNO	PARTNO	PRICE
51	221	0.30
51	231	0.10
53	241	0.08
61	241	0.05

Figura 10. Sottointerrogazioni correlate

La correlazione tra le due interrogazioni e' realizzata tramite una Label (X nel nostro esempio, max 18 caratteri) la quale consente di calcolare il doppio del minimo prezzo per ogni numero di pezzo preso in considerazione per la selezione nell'interrogazione principale.

Concettualmente la query e' valutata nel seguente modo:

1. Dalla tabella Quotations viene ricavata una tabella, identificata con la variabile di correlazione X e chiamata tabella X.

2. Il sistema identifica X.Partno con la tabella X e usa i valori in quella colonna per valutare la query. (L'intera query e' valutata una volta per ogni Partno nella tabella X).

2.8 Uso delle viste per semplificare le interrogazioni

Il sistema SQL/DS fornisce la possibilita' di memorizzare oltre a tabelle e indici particolari oggetti chiamati "view". Una view puo' essere considerata una tabella logica (virtuale) derivata da una o piu' tabelle, da altre view o combinazioni di view e tabelle. Il sistema presenta il risultato di una view sotto forma di tabella. Come tale sara' costituita da righe e colonne ma sulla quale alcune operazioni, permesse sulle tabelle reali, in questo caso non sono valide. Ad esempio non e' possibile creare un index su una view in quanto non esiste fisicamente oppure non e' possibile aggiungere delle colonne ecc...
Invece altri comando (Select) possono esser scritti sulle view esattamente come se fossero delle tabelle reali. Allorche' viene lanciata una operazione di query su una view, il sistema combina la query con la definizione della view per produrre una nuova query per le tabelle reali su cui e' definita la view. A questo punto il sistema processa la query nel modo usuale. Per esempio supponiamo di definire la seguente view:

```
create view manuale (mfr,part,days) as -
       select suppno,partno,delivery_time -
       from quotations where delivery_time < 10
```

e inviare successivamente il comando:

```
select  part,days -
 from manuale     -
 where  mfr=70  -
 order by 2
```

PART	DAYS
316	7
315	8

Figura 11. Interrogazione di una view

dunque SQL/DS combina questa query con la definizione della view ed elabora la query risultante:

```
select  partno,delivery_time -
  from quotations  -
  where  delivery_time < 10 -
  and suppno=70  -
  order by 2
```

Vediamo un esempio in cui e' particolarmente utile definire una view. Supponiamo di voler eseguire una interrogazione abbastanza complessa ed eseguire a fronte di quella risposta altre interrogazioni. Cioe' avere accesso ad una tabella contenente i risultati di una interrogazione precedente.

Tramite il comando :

```
create view  quotes  -
  as select i.partno,description,price -
  from inventory i, quotations q -
  where i.partno=q.partno  -
  and  q.suppno=54
```

viene memorizzato lo statement Select come definizione di una view. Lo statement fornisce il nome della view (Quotes) mentre sono opzionali i nomi di colonna. In questo caso i nomi di colonna non sono forniti ed il sistema prende i nomi delle colonne da cui sono derivate. La view e' dunque formata dalle colonne Partno, Description e Price per il fornitore 54. A questo punto tramite semplici comandi di interrogazione si puo' andare a visualizzare il contenuto. Per esempio:

```
select  * -
  from quotes  -
  order by partno
```

fornisce il risultato

PARTNO	DESCRIPTION	PRICE
209	CAM	18.00
221	BOLT	0.10
231	NUT	0.04
241	WASHER	0.02

Figura 12. Interrogazione di una seconda view

2.9 Denominazione di colonne in una view

Se alcune colonne della view non sono ottenute direttamente da un campo di dati; ovvero se una colonna della view contiene valori che vengono calcolati attraverso un'espressione o una funzione implicita, e' necessario fornire nuovi nomi di colonna della vista. Le colonne cosi' ottenute si chiamano anche colonne virtuali in quanto contengono dati virtuali.
Bisogna specificare nomi nuovi di colonne anche nel caso in cui i campi selezionati della view non hanno nomi univoci (per esempio la view e' un Join di due tabelle ognuna delle quali ha una colonna con lo stesso nome).
Per esempio vogliamo creare nella view di nome Quotes una colonna con l'importo di ciascun ordine dato da Price*Qonorder. Non potendo usare questo nome come nome di colonna della view in quanto verrebbe interpretato come prodotto della colonna Price e della colonna Qonorder e quindi in modo errato e' necessario fornire un nuovo nome. La forma del comando e' percio':

```
create view cost (parto,description,invoice_cost) -
 as select i.partno,description,price*qonorder -
 from inventory i,quotations q -
 where i.partno=q.partno -
  and q.suppno=54
```

Viene cosi' creata una view di nome Cost che prevede le colonne di nome Partno, Description e Invoice_Cost per il fornitore 54. Per visualizzare la vista sara' sufficiente dare il comando:

```
select * -
  from cost -
  order by partno
```

PARTNO	DESCRIPTION	INVOICE_COST
209	CAM	0.00
221	BOLT	15.00
231	NUT	8.00
241	WASHER	4.00

Figura 13. Interrogazione di una terza view

3.0 STRUTTURA LOGICA DI UN DATA BASE

Il data base SQL/DS e' un insieme di tabelle, indici e informazioni di supporto dei dati mantenuti dal sistema. Le informazioni di supporto sono relative a informazioni di controllo per esempio come una tabella e' formattata e dove e' memorizzata; e informazioni relative al recovery dei dati ovvero restore dei dati ad un precedente stato.

Il data base SQL/DS e' composto di:

- uno o piu' Storage Pools
- uno o piu' Data Base Spaces (DBSPACES)
- un data set di Directory
- uno o piu' data sets di Log

Mentre gli argomenti Directory e data sets di Log hanno un maggior impatto per la gestione del sistema e quindi ne tralasciamo l' esposizione; i concetti DBSPACE e Storage Pools, per una migliore comprensione della logica di sistema, possono essere di maggiore utilita' per l' utente.

Un DBSPACE definito durante la generazione del sistema o tramite una facility che consente di aggiungere dinamicamente, e' un insieme logico dove sono raggruppati i dati di utente. La dimensione di un DBSPACE e' multipla di 128 pagine; quindi 128, 256 o 384 ecc. dove ogni pagina e' formata da 4k bytes di memoria. Quando viene creata una tabella, lo user puo' specificare il nome del proprio DBSPACE a cui e' associata la tabella oppure lasciare al sistema il compito di assegnare la tabella ad un DBSPACE privato.
In ogni modo un DBSPACE non e' uno spazio fisico; una reale allocazione di spazio su disco ma una allocazione di "page tables" nella Directory del sistema. Il sistema alloca dinamicamente spazio su disco per supportare pagine di un DBSPACE in relazione ad una richiesta. Quindi un DBSPACE vuoto non occupa spazio reale su disco nei data sets definiti per supportare i DBSPACES ed analogamente pagine non usate non occupano spazio. Dunque un DBSPACE rappresenta uno spazio che e' mappato a memoria reale su disco solo quando le tabelle e indici sono creati e quando sono inseriti i dati. Si puo' cosi' pensare a un DBSPACE come uno spazio potenziale di disco limitato dalla sua dimensione allorche' viene definito.
Spazio su disco per un DBSPACE e' automaticamente ottenuto dal sistema dai suoi minidischi per supportare la dimensione reale delle tabelle e indici nel DBSPACE.

A fronte di ogni DBSPACE ci sono da 1 a 8 pagine di "Header" che contengono informazioni di controllo sulle tabelle e indici memorizzati nel DBSPACE; mentre le righe di una tabella sono memorizzate in pagine di dati immediatamente dopo le pagine di Header all'inizio del DBSPACE. Gli entry di indice sono memorizzate a partire dalla fine del DBSPACE e pagine non usate se esistono, sono nel mezzo.

Figura 14. Tabella ed indici memorizzati in un DBSPACE

Un DBSPACE puo' ovviamente contenere piu' di una tabella e piu' indici su ogni tabella ma il numero totale di tabelle e indici non puo' eccedere 255 per ogni DBSPACE.

Si possono avere 3 tipi di DBSPACE:

- pubblico: usato per dati che debbono essere condivisi da piu' di uno user nello stesso tempo.
- privato: per dati che debbono essere acceduti da uno user alla volta.
- internal: sono DBSPACE temporanei usati dal sistema in elaborazioni interne.

Allorche' viene definito un DBSPACE esso e' assegnato a uno Storage Pool cioe' ad un Pool di blocchi di memoria di 4K su disco chiamati Slots attraverso i quali viene gestita l'allocazione di spazio sui Dasd assegnati alla banca dati. Le aree di disco sono definite in termini di minidisco.
Si possono definire piu' Storage Pool (da 1 a 999) ognuno dei quali e' composto da uno o piu' DBEXTENT. I DBEXTENT sono il mezzo fisico in cui i dati vengono memorizzati ed appaiono internamente al sistema come uno spazio contiguo su disco e usato in unita' di pagine di 4K bytes.

Le varie unita' definite possono esser messe in queste relazioni:

- un data base e' fatto di uno o piu' Storage Pools
- uno Storage Pool e' fatto di uno piu' DBEXTENT fisici
- uno Storage Pool supporta uno o piu' DBSPACE logici

- i dati di un DBSPACE debbono stare in uno Storage Pool ma un DBSPACE puo' condividere piu' DBEXTENT
- una tabella e i suoi indici debbono stare in un DBSPACE il quale puo' contenere fino a 255 tabelle con i loro indici.

Volendo rappresentare i concetti esposti fornendo al tempo stesso un livello di gerarchia si ha:

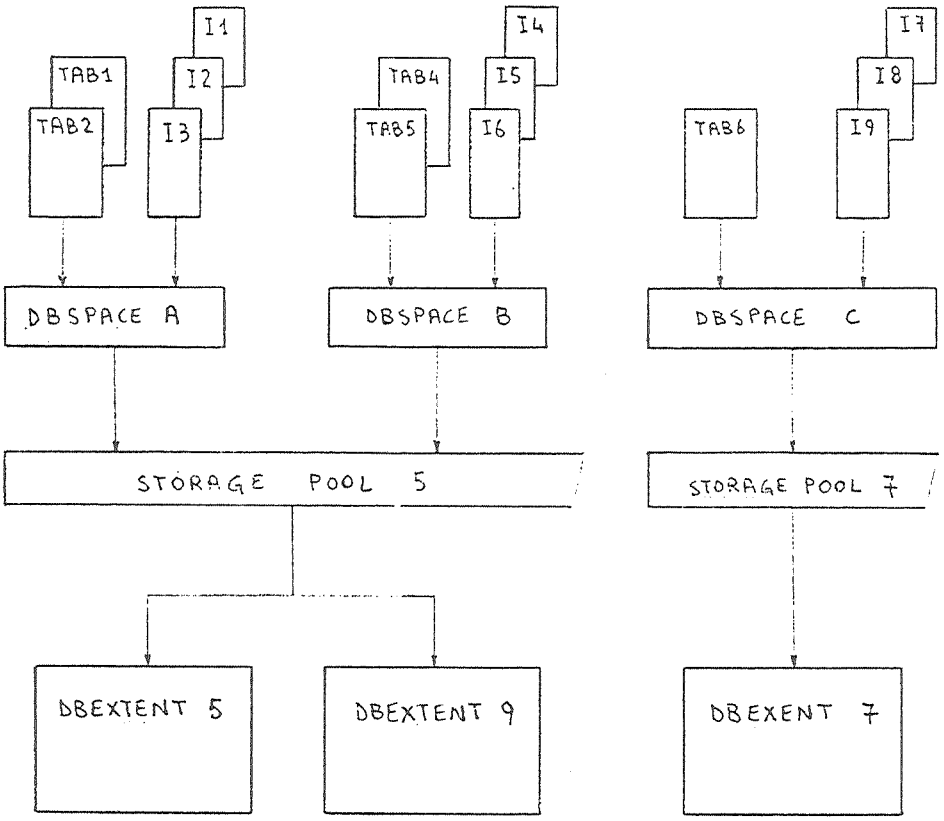

Figura 15. Rappresentazione logica di un Data Base

4.0 GESTIONE DELLE PROPRIE TABELLE

Dalla definizione di DBSPACE ne consegue che l'unico oggetto su cui ciascun utente ha facolta' di lavorare in modo autonomo sono le proprie tabelle sia che appartengono al DBSPACE privato dell'utente oppure ad un DBSPACE di tipo pubblico.
Vediamo come gestire le proprie tabelle, come usarle in comune con altri utenti e come migliorare le prestazioni delle interrogazioni.

La gestione delle proprie tabelle comporta:

- creazione di tabelle
- decisione su dove devono essere memorizzate
- identificazione del contenuto minimo di una tabella
- cancellazione di tabelle
- copiatura di dati da una tabella all'altra
- aggiunta di colonne a una tabella

4.1 Creazione di proprie tabelle

Per creare una tabella e' necessario fornire il nome della tabella stessa, i nomi delle colonne di cui la tabella e' costituita ed il tipo di dati che ciascuna colonna dovra' contenere.
Vogliamo creare una tabella su cui registrare informazioni trimestrali di inventario formata da 3 colonne. La prima colonna (Partno) che contiene i numeri di parte e quindi piccoli numeri interi viene definita SMALLINT. La seconda colonna di nome (Quarter) identifica il trimestre dell'anno al quale si riferisce l'inventario e sara' quindi una stringa alfanumerica di 4 caratteri. La terza colonna (Qinv) contenente la quantita' disponibile al termine del trimestre e' definita INTEGER perche' dovra' contenere grandi numeri interi.

Il comando e':

```
create table  inv  -
              (partno  smallint, -
               quarter char(4), -
               qinv    integer)
```

eseguito il comando e creata la tabella i dati possono essere inseriti nei modi gia' noti oppure attraverso l'utility Data Base Service.

4.2 Dove vengono memorizzate le tabelle

Abbiamo detto che un DBSPACE e' una porzione del Data Base in cui sono memorizzate le tabelle ed i loro indici. Dopo aver definito il DBSPACE nella Directory del sistema per poterlo usare e' necessario dare il comando ACQUIRE DBSPACE in cui sono forniti alcuni parametri per la definizione delle sue caratteristiche. A questo punto il DBSPACE e' reso disponibile all'utente che lo ha definito.
In particolare se il tipo di DBSPACE e' Privato lo User che ha fatto Acquire ne e' proprietario; se invece e' Pubblico allora il DBSPACE e' posseduto dal sistema e reso disponibile a tutti gli utenti che lo vorranno usare. E' chiaro dunque come un utente non puo' creare tabelle in un DBSPACE privato posseduto da altri utenti.

Un DBSPACE privato e' destinato a contenere le vostre tabelle che devono essere usate da parecchi utenti contemporaneamente. Le tabelle in un DBSPACE privato potranno essere gestite; inserimento, cancellazione o aggiornamento di dati, solamente dal proprietario del DBSPACE a meno che ad altri utenti viene concesso di fare queste operazioni. In ogni caso mentre vengono fatte tali operazioni nessun utente puo' accedere le tabelle.
Se invece le tabelle debbono essere oggetto di aggiornamenti, cancellazioni o selezioni da parecchi utenti, devono essere memorizzate in un DBSPACE di tipo Pubblico. In questo caso una tabella puo' essere aggiornata contemporaneamente da parecchi utenti (in ogni caso due utenti non possono usare la stessa riga contemporaneamente).

E' possibile specificare al momento in cui una tabella e' creata in quale DBSPACE dovra' essere memorizzata. Supponiamo di avere due DB Space riservati con nome Cnucel e Cnuce2. Per inserire la tabella Inv nel DB Space denominato Cnucel emettere il comando:

```
create table  inv  -
       (partno   smallint, -
        quarter  char(4),  -
        qinv     integer)  -
    in cnucel
```

4.3 Identificazione del contenuto minimo di una tabella

Al momento in cui viene creata una tabella e' possibile disporre che i campi chiave contengano sempre valori validi (non nulli). Tramite l'opzione Not Null nella definizione della colonna del comando Create viene impedito l'uso di valori nulli.

```
create table inv -
   (partno  smallint  not null, -
    quarter char(4)   not null, -
    qinv    integer)
```

cioe' le colonne Partno e Quarter non possono avere valore nullo mentre nella colonna Qinv sono ammessi valori nulli. Lo strumento e' particolarmente utile allorche' vengono copiati dati da altre

tabelle. L'opzione Not Null in questo caso impedisce l'inserimento nella tabella di righe incomplete.

4.4 Eliminazione di una tabella

Per eliminare una tabella dal data base il comando e' Drop Table. Nel nostro esempio:

```
drop table inv
```

elimina tutte le righe della tabella Inventory e la definizione della tabella stessa. Per cancellare soltanto le righe (ma non la tabella) vedere il comando Delete.

4.5 Copiatura di dati da altre tabelle

E' possibile caricare i dati in una tabella copiandoli da un'altra. Il comando:

```
insert into inv (partno,quarter,qinv) -
     select partno,'4q81',qonhand  -
       from inventory
```

carica i dati dalla tabella Inventory alla tabella Inv. Cioe' vengono copiati i numeri parte e la quantita' disponibile per ciascuna riga della tabella Inventory nella colonna Partno e Qinv della tabella Inv ed inserito il valore '4q81' nella colonna Quarter per ciascuna riga.

4.6 Aggiunta di una colonna a una tabella

Creata una tabella puo' esser utile aggiungere successivamente colonne alla tabella che al momento della creazione non erano previste. Per esempio aggiungere la colonna di nome Asset alla tabella Inv per inserire informazioni contabili:

```
alter table inv -
    add asset decimal (9,2)
```

quindi con il comando Update vengono introdotti i dati.

4.7 Uso di proprie tabelle in comune con altri utenti

Abbiamo visto che le tabelle create da un utente sono da questo possedute e solamente lui le puo' usare. Possono esserci dei casi in cui queste tabelle sono di interesse per altri utenti e quindi si pone il problema di come renderle disponibili.
Per consentire ad altri utenti l'accesso alle proprie tabelle deve esser dato il comando di Grant mentre con il comando Revoke viene annullata tale autorizzazione.

4.7.1 COME CONCEDERE AD ALTRI DI SELEZIONARE LE VOSTRE TABELLE

L'operazione piu' comune eseguita su una tabella e' consultare i dati quindi, per consentire comandi di Select sulle proprie

tabelle dovra' essere specificato nel comando Grant. Vogliamo consentire di leggere la tabella Inv ad un utente di nome User1; il comando relativo e':

```
grant   select  -
   on   inv   -
   to   user1
```

4.7.2 CONCESSIONE DI ALTRI PRIVILEGI SULLE VOSTRE TABELLE

Oltre ad operazioni di Select e' possibile concedere agli utenti anche altre operazioni (Insert, Update e Delete). Per esempio per concedere la capacita' di Insert e Update sulla tabella Inv all'utente User1 il comando e':

```
grant  insert,update -
     on   inv   -
     to   user1
```

In questo modo l'utente User1 sara' autorizzato ad aggiungere righe alla tabella Inv o aggiungere una o piu' colonne alla tabella. Mentre non avendo autorizzato il comando Delete, l'utente User1 non potra' cancellare righe dalla tabella.

4.7.3 CONCESSIONE DI TUTTI I PRIVILEGI AD UN UTENTE

Per poter concedere tutti i privilegi su una tabella anziche' elencarli bastera' specificare la parola All:

```
grant all  -
  on   inv -
  to   user1
```

4.7.4 CONCESSIONE DI PRIVILEGI A PIU' UTENTI

Volendo concedere la capacita' di Update ad un gruppo di utenti (User1, User2, User3) il comando sara':

```
grant update -
   on inv -
   to user1,user2,user3
```

4.7.5 REVOCA DEI PRIVILEGI CONCESSI

Tramite il comando Revoke vengono revocati i privilegi concessi attraverso il comando Grant. Vogliamo togliere la possibilita' all'utente User1 di fare Update sulla tabella Inv. Il comando sara':

```
revoke update -
    on inv  -
  from user1
```

Analogamente al comando Grant possono essere identificati utenti multipli e privilegi multipli.

4.8 Accesso alle tabelle che appartengono ad altri utenti

Quando in un comando SQL fate riferimento ad una tabella, SQL/DS presume che si tratti di una tabella che vi appartiene. Volendo accedere ad una tabella o ad una view che appartiene ad un altro utente e' necessario identificare l'utente nonche' il nome della tabella o della view. Cio' avviene inserendo l'userid davanti al nome della tabella (o della view) e separando i due nomi con un punto.

Per accedere la tabella Inventory posseduta dal sistema il comando sara':

```
select * -
  from sqldba.inventory
```

Analogamente altri utenti che intendono usare vostre tabelle devono ricorrere a questo metodo. La necessita' di aggiungere l'userid per le tabelle o le view di proprieta' di altri utenti consente di usare nomi gia' utilizzati da altri. Per contro puo' essere piuttosto scomodo fornire costantemente un userid per la tabella o la view di un altro utente che usate frequentemente. Potete evitarlo creando sinonimi per le tabelle o le view di proprieta' di altri. Per esempio, potete assegnare un sinonimo da usare in luogo di Sqldba.Inventory emettendo:

```
create  synonym  dbainv  -
   for  sqldba.inventory
```

Per fare riferimento a Sqldba.Inventory bastera' scrivere:

```
select  * -
  from dbainv
```

ed il sistema tradurra' il sinonimo in Sqldba.Inventory dandovi gli stessi risultati della interrogazione precedente.

4.9 Indicizzazione di una tabella

Definita una tabella e' possibile creare sulla stessa uno o piu' indici. L' indice e' una struttura che consente di aumentare la velocita' di risposta del sistema a fronte di operazioni di query. Il sistema SQL/DS, per trovare le righe di una tabella in cui non sono stati creati indici, deve fare lo scan di tutte le pagine nelle tabelle del DBSPACE con conseguenti limiti alle prestazioni. Percio' ogni tabella dovrebbe avere almeno un indice a meno che la tabella sia la sola nel DBSPACE. Anche in questo caso, un indice e' desiderabile per aumentare la velocita' di accesso alla tabella.
Un indice puo' esser definito su piu' colonne di una tabella (max 16) ma non puo' esser definito su piu' tabelle.

Come esempio consideriamo una tabella contenente i dati sugli impiegati di una compagnia. E' desiderabile avere alcuni indici su questa tabella per utilizzare questa in modo piu' efficace. Per esempio:

- un indice sulla colonna del nome degli impiegati rendera' piu' efficace la ricerca dei dati di uno specifico impiegato;
- un indice sulla colonna degli indirizzi aiutera' un programma che ha creato label di posta raggruppate per destinazione;
- un indice sulle colonne tipo di lavoro e salario rendera' piu' veloce l'analisi fatta dal personale del dipartimento.

Gli indici possono essere creati o cancellati dinamicamente tramite ISQL, il DBS utility o applicazione scritta dallo user. Creazione e cancellazione di indici puo' essere fatta mentre altri stanno usando SQL/DS.

Volendo creare un indice di nome "invdescript" sulla tabella Inventory per la colonna Description in quanto ogni ricerca sulla tabella viene fatta tramite la descrizione il comando sara':

```
create  index  invdescript -
   on inventory   -
     (description)
```

5.0 CREAZIONE ED USO DELLE ROUTINES

Allorche' e' necessario eseguire comandi SQL complicati, abbiamo visto come attraverso la tecnica di memorizzazione, il lavoro possa essere semplificato.

Una estensione della memorizzazione dei comandi SQL e' il concetto di "routines". Una routine e' una sequenza costituita da uno o piu' comandi memorizzati sotto un nome di identificazione. Un utente che ha necessita' di eseguire in modo ripetitivo certi comandi puo' creare e memorizzare questi comandi in una routine. Quando la routine viene eseguita, ciascun comando viene elaborato come se fosse stato impostato dalla tastiera.
Un altro vantaggio della memorizzazione delle query e' che lo user non familiare con il linguaggio SQL usando le routines si trova facilitato.

Infine analogamente ad un comando SQL memorizzato, una routine puo' contenere campi variabili.

5.1 Come stabilire dove sono memorizzate le routines.

Le routines di ogni user risiedono in una tabella speciale chiamata "Routine". Ogni utente puo' definire una propria tabella costituita da 4 colonne:

```
create table routine -
        (name    char(8)    not null, -
         seqno smallint    not null, -
         command varchar(69) not null, -
         remarks varchar(40))
```

dove:

- **name** - identifica le righe che appartengono ad una particolare routine;
- **seqno** - specifica la sequenza in cui i comandi della routine vengono eseguiti;
- **command** - e' la colonna in cui vengono inseriti i comandi SQL e ISQL;
- **remarks** - questa colonna puo' essere usata per riportare alcune note sulla routine.

5.2 Memorizzazione di una routine

Creata la tabella di nome Routine, attraverso i comandi SQL Insert o ISQL Input, vengono memorizzate le procedure. Analogamente ad una qualsiasi tabella si possono eseguire operazioni di aggiornamento, cancellazione ed inserimento.

5.3 Esecuzione di una routine

Supponiamo di aver creato una routine di nome Qreport cosi' definita:

```
NAME      SEQNO   COMMAND                                  REMARKS
--------  -----   -------------------------------          ---------
QREPORT      10   select partno,suppno,qonorder -
QREPORT      20   from quotations -
QREPORT      30   where partno in (&1,&2,&3) -
QREPORT      40   order by partno,suppno -
QREPORT      50   format group partno
QREPORT      60   format subtotal qonorder
QREPORT      70   format ttitle 'parts on order'
QREPORT      80   print
QREPORT      90   end
```

Figura 16. Routine Profile n.1

La procedura cosi' definita viene mandata in esecuzione con il comando Run:

```
run qreport (221 222 231)
```

e crea un prospetto delle quantita' in ordine per i pezzi 221, 222 e 231.

5.4 Uso di comandi Select in una routine

Nell'ambito della definizione di una routine e' opportuno ricordare come il comando Select inserito nella routine, non esegue automaticamente la visualizzazione dei risultati sul terminale. Per poter eseguire cio' e' necessario inserire nella routine, nella posizione desiderata il comando Display. Volendo aggiornare la routine Qreport emettete il comando:

```
insert into routine (name,seqno,command) -
   values ('qreport',75,'display')
```

e la routine diventa:

```
NAME      SEQNO   COMMAND                                  REMARKS
--------  -----   -------------------------------  ---------
QREPORT      10   select partno,suppno,qonorder -
QREPORT      20   from quotations -
QREPORT      30   where partno in (&1,&2,&3) -
QREPORT      40   order by partno,suppno -
QREPORT      50   format group partno
QREPORT      60   format subtotal qonorder
QREPORT      70   format ttitle 'parts on order'
QREPORT      80   print
QREPORT      90   end
QREPORT      75   display
```

Figura 17. Routine Profile n.2

Eseguendo la routine tramite il comando Run il risultato della interrogazione verra' visualizzato sul terminale per eventauli modifiche.

5.5 Le routines Profile

Una routine di nome Profile viene eseguita automaticamente ogni volta che e' attivato ISQL consentendo di stabilire le caratteristiche operative della sessione a terminale. In questo tipo di routine non sono ammesse variabili posizionali in quanto non c'e' modo di trasmettere parametri essendo eseguita in modo automatico.
Nel caso in cui esiste una routine Profile principale, valida per tutti gli utenti, viene eseguita prima della propria routine. In questo modo se la routine principale imposta delle caratteristiche operative di lavoro queste potranno essere modificate e personalizzate da ciascun utente tramite la propria routine di Profile.

6.1 Tabella Inventory

PARTNO	DESCRIPTION	QONHAND
207	GEAR	75
209	CAM	50
221	BOLT	650
222	BOLT	1250
231	NUT	700
232	NUT	1100
241	WASHER	6000
285	WHEEL	350
295	BELT	85
301	FUSE	300
302	FUSE	367
310	PIN	150
311	PIN	292
315	SWITCH	421
316	SWITCH	342
320	RELAY	0
321	RELAY	0
322	RELAY	0
323	RELAY	0

6.2 Tabella Suppliers

SUPPNO	NAME	ADDRESS
51	DEFECTO PARTS	16 BUM ST., BROKEN HAND, WY
52	VESUVIUS, INC.	512 ANCIENT BLVD., POMPEII NY
53	ATLANTIS CO.	8 OCEAN AVE., WASHINGTON DC
54	TITANIC PARTS	32 LARGE ST., BIGTOWN, TX
57	EAGLE HARDWARE	64 TRANQUILITY PLACE, APOLLO MN
61	SKY PARTS	128 ORBIT BLVD., SIDNEY AUSTRALIA
64	KNIGHT LTD.	256 ARTHUR COURT, CAMELOT ENGLAND
70	COMPANY1	50 GRAND AVE, SPRINGFIELD ILL
71	COMPANY2	130 OAK HILL AVE, DETROIT MI
72	COMPANY3	241 OLD CASTLE RD, PHILADELPHIA PA

6.3 Tabella Quotations

SUPPNO	PARTNO	PRICE	DELIVERY_TIME	QONORDER
51	221	0.30	10	50
51	231	0.10	10	0
53	222	0.25	15	0
53	232	0.10	15	200
53	241	0.08	15	0
54	209	18.00	21	0
54	221	0.10	30	150
54	231	0.04	30	200
54	241	0.02	30	200
57	285	21.00	14	0
57	295	8.50	21	24
61	221	0.20	21	0
61	222	0.20	21	200
61	241	0.05	21	0
64	207	29.00	14	20
64	209	19.50	7	7
70	301	0.20	15	100
70	302	0.21	16	0
70	310	40.90	30	0
70	311	41.00	31	100
70	315	1.00	8	0
70	316	1.01	7	0
70	320	3.00	18	0
70	321	3.10	17	0
70	322	3.15	16	0
71	301	0.25	10	0
71	302	0.26	11	200
71	310	55.00	20	0
71	311	56.00	21	0
71	315	1.10	3	65
71	316	1.11	16	0
71	320	3.20	18	0
71	321	3.19	19	0
71	322	3.21	20	0
72	301	0.15	20	0
72	302	0.16	21	0
72	310	43.10	23	300
72	311	48.00	24	0
72	315	1.49	15	0
72	316	1.50	14	850
72	320	3.50	11	0
72	321	3.51	12	0
72	322	3.49	13	0
72	323	3.52	14	0

BIBLIOGRAFIA

SQL/DS: General Information for VM/SP IBM GH24-5064

SQL/DS: Terminal User's Reference for VM/SP IBM SH24-5067

SQL/DS: SQL/DS Terminal User's Guide for VM/SP IBM SH24-5045

SQL/DS: SQL/DS Concepts and Facilities for VM/SP IBM GH24-506

BIBLIOGRAFIA

SQL/DS: General Information for VM/SP IBM GH24-5064

SQL/DS: Terminal User's Reference for VM/SP IBM SH24-5067

SQL/DS: SQL/DS Terminal User's Guide for VM/SP IBM SH24-5045

SQL/DS: SQL/DS Concepts and Facilities for VM/SP IBM GH24-506