Le basi del
Data Base

Data Base Management System: il modello reticolare dei dati

di Andrea de Prisco

L'argomento di questo mese riguarda i sistemi di gestione per basi di dati di tipo reticolare. Nati da una proposta del Data Base Task Group del Codasyl (Conference on data systems languages) per sopperire alle limitazioni proprie del modello gerarchico, pur senza l'appoggio della IBM si sono notevolmente diffusi nell'ultimo decennio.

I sistemi reticolari

La caratteristica principale del modello reticolare dei dati è quella di strutturare una base come un grafo più che come gli alberi di definizione visti lo scorso mese per il modello gerarchico. Un grafo, in generale, e quella di figura 1: vediamo dei nodi, contrassegnati da lettere, e degli archi tra questi. In generale, anche se bisognerebbe parlare di multigrafi, più archi possono collegare due nodi qualsiasi. Va fatto notare che il grafo non è qualcosa che riguarda solo basi di dati, ma un normalissimo strumento matematico usato in più campi per visualizzare particolari situazioni: basta volta per volta mettersi d'accordo su cosa è un nodo e cosa rappresentano gli archi. Nel caso delle basi di dati, ogni nodo denota un insieme di dati e (indovinate un po') gli archi tra questi le associazioni tra dati. La direzione della freccia è importante come vedremo in seguito in quanto, di fatto, le associazioni sono orientate: vanno da un particolare insieme a un altro.

Il meccanismo delle associazioni tra dati si realizza tramite delle classi dette appunto associazione, generalmente non mostrate nel grafo per non complicarlo ulteriormente: basta la freccia.

Le associazioni vanno tutte da una classe detta Padre (o dominio) a una classe detta Figlia (o codominio). Resta inteso che una classe Figlia (relativamente a una associazione) può essere a sua volta Padre di un'altra classe (relativamente a un'altra associazione). Ad esempio, tornando alla figura 1 (posto che i nodi sono insiemi) la classe C è padre della classe D e figlia della classe B.

La distinzione tra domini e codomini, è dovuta al fatto che le associazioni hanno tutte diretta multipla e inversa univoca: ogni elemento della classe Padre può essere in associazione con più elementi della classe Figlia, ma gli elementi della classe Figlia sono in associazione con un solo elemento della classe Padre. Per ricordarselo basta pensare ai propri figli: "dunque... io ho più figli.... i miei figli hanno un solo padre (!)".

In figura 2 è mostrato un pezzettino di base di dati: l'associazione HaInPrestito dalla classe Utenti alla classe Materiali (toh! la biblioteca!). Il vincolo implicito del tipo di associazione non è violato: ogni utente può avere più libri in prestito, ogni libro prestato può stare al più presso un solo utente.

Le associazioni si realizzano creando una classe supplementare (detta classe associazione) che contiene elementi del tipo:

```
(a1, [b1, b2, b3,..., bn])
```

dove a1 è un puntatore alla classe dominio, [b1,...,bn] una sequenza di puntatori alla classe codominio. C'è un elemento di questo tipo per ogni elemento della classe Padre, eventualmente con la sua sequenza vuota se non è in associazione con alcun "figlio" (un utente che non ha, al momento

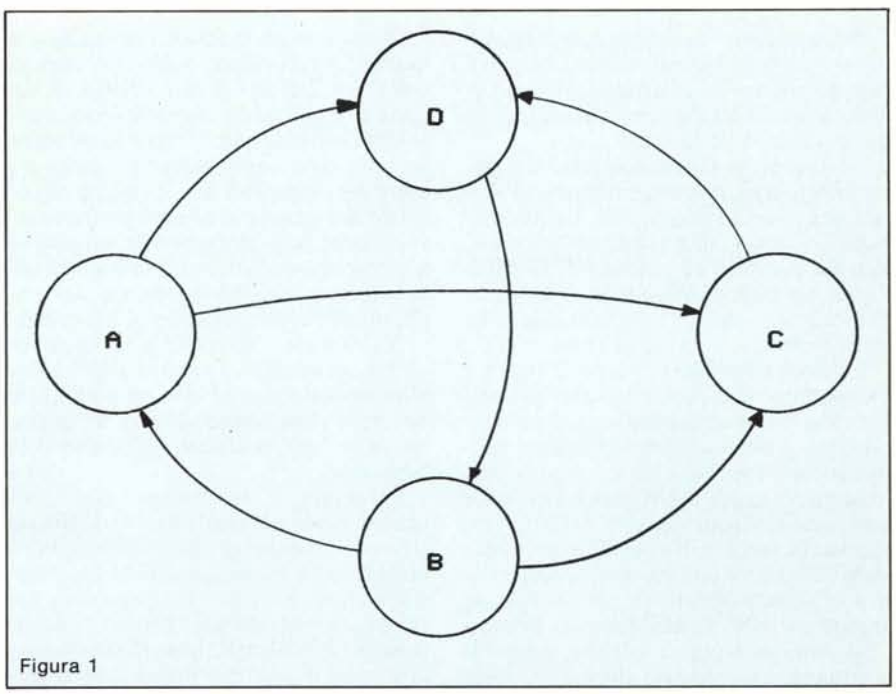

Figura 1

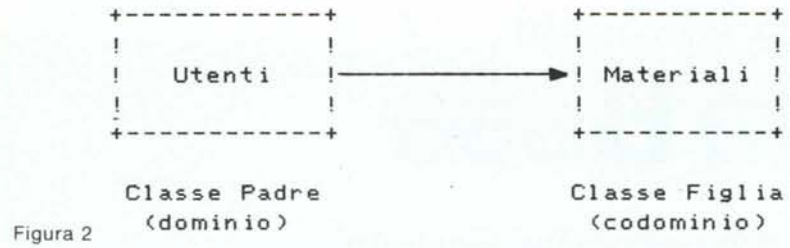

Figura 2

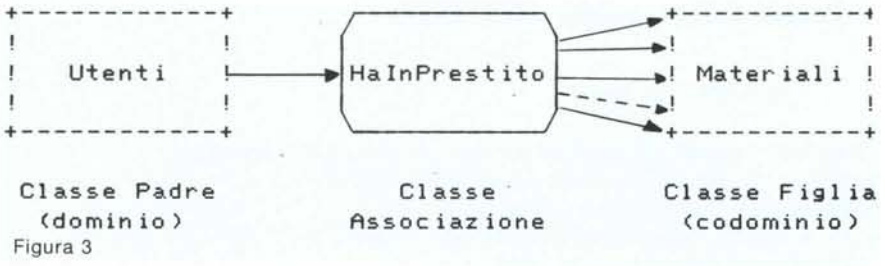

Figura 3

attuale, alcun libro in prestito). Il puntatore potrebbe, ad esempio, essere il numero d'ordine di inserimento nella base di dati.

In figura 3 è mostrata l'associazione di figura 2, come effettivamente realizzata. La classe al centro, come detto, contiene tutti gli elementi del tipo visto sopra. È chiaro che non tutti gli elementi del dominio hanno lo stesso numero di figli, e ciò giustifica l'imprecisato numero di frecce con l'insieme codominio.

La figura 4 mostra un particolare: l'utente Aurelio Tinti ha in prestito I Promessi Sposi, l'Eneide e la Divina Commedia. L'utente è la registrazione n. 492, i tre testi, rispettivamente, 65, 321 e 142. L'elemento centrale, che realizza la correlazione, contiene semplicemente i 4 numeri (per l'esattezza il primo numero e gli altri 3).

Inversa totale o parziale

Per non complicare troppo le cose, abbiamo volutamente tralasciato un particolare: nei sistemi reticolari esistono due tipi di associazioni tra dati, fermo restando che quanto detto vale in tutti i casi.

La distinzione riguarda la totalità o meno dell'inversa, in parole tecniche la surgettività o la non surgettività. Un'associazione tra dati si dice totale se "colpisce" uno per uno tutti gli elementi della classe Figlia; parziale o non totale se esistono elementi nella classe Figlia non colpiti da alcun elemento nella classe Padre.

L'associazione HaInPrestito di figura 2 è un esempio di associazione non surgettiva: è ovvio che nella biblioteca ci saranno libri non prestati ad alcuno (presenti realmente tra gli scaffali). Se la classe colpita fosse invece quella dei Materiali Prestati (e non, indifferentemente, di tutti i libri), l'associazione sarebbe di tipo surgettivo: ogni libro della classe sarà (necessariamente) figlio di qualche utente. In altre parole, se appartiene alla classe Materiali Prestati vuol dire che è presso qualche utente. Il sistema stesso assicurerà che non esistono elementi di classi Figlia, surgettivamente in relazione con classi Padre, inversamente non associati con alcun elemento.

Un'altra limitazione (ben) imposta dalle associazioni surgettive è che non è possibile distruggere un elemento della classe Padre senza distruggere anche gli elementi nella classe Figlia ad esso correlati. I sistemi reticolari in commercio, a seconda di libere scelte di progetto, si comportano in casi del genere in due diversi modi. Se distruggiamo un elemento di una classe Padre, in associazione surgettiva con altri elementi di un'altra classe, o il sistema abortisce l'operazione segnalando un messaggio di errore o automaticamente distrugge anche i figli, eventualmente facendo una vera e propria strage ricorsivamente anche a livelli inferiori (figli dei figli ecc.).

La definizione della base

Quando vogliamo costruire una base di dati di tipo reticolare, dobbiamo innanzitutto ben definire il suo schema: il suo grafo di definizione. Dobbiamo cioè avere le idee chiare su tutte le classi da adoperare, sulle varie associazioni (non senza stabilire per ognuna di esse il vincolo di surgettività o meno) e conoscere la struttura, i vari campi, degli elementi delle varie classi. La struttura degli elementi non avrà espliciti riferimenti a associazioni, ma sarà semplicemente una lista di campi o attributi.

Ad esempio, un utente avrà un nome, un'età, un recapito, un numero telefonico e altre informazioni ad esso relative. Un libro della classe materiali avrà un autore, un titolo, una posizione all'interno della biblioteca.

Per definire le associazioni si dovrà indicare un nome e le classi Padre e Figlia che riguarda. Sempre in merito alla figura 2 HaInPrestito è un'associazione tra Utenti e Materiali. Nel caso di associazioni non surgettive basta questo. Quando un libro è prestato a qualcuno, basterà comunicare al sistema di costruire l'associazione semplicemente dicendo: Utente Pippo HaInPrestito Biancaneve e i sette nani. Da questo momento potremo ad esempio cercare l'utente Pippo e conoscere quali libri ha in prestito così come cercare il libro di Biancaneve e risalire da questo all'utente che se n'è impossessato. Questo è un classico esempio di navigazione nella base: da una classe si passa ad un'altra sfruttando le associazioni tra dati.

Analizziamo ora il caso in cui l'associazione è surgettiva, ad esempio tra la classe Utenti e la classe MaterialiPrestati. È ovvio che, rispetto al caso precedente, gli utenti come entità del mondo osservato (!) non cambiano; diversa è la situazione per i materiali: se cataloghiamo un libro in prestito, oltre a Autore e Titolo, un'altra informazione importante è il nome dell'utente al quale è stato consegnato. Gli elementi di MaterialiPrestati avranno in tutto tre campi: Autore, Titolo e Utente. Le associazioni surgettive sono dette anche automatiche essendo automatica la correlazione tra dati. È chiaro che tutti i libri prestati che hanno nel campo utente Tizio sono in prestito all'utente Tizio.

Per definire un'associazione surgettiva si indicano le classi che essa coinvolge e quale condizione gli elementi devono soddisfare per essere tra loro in relazione. Ad esempio, sempre nella definizione scriveremo qualcosa del genere:

HaInPrestito automatic set Utenti → MaterialiPrestati on Nome = Utente.

HaInPrestito è il nome della associazione; automatic set indica che l'associazione è surgettiva (set in inglese vuol dire anche associazione); seguono le classi coinvolte e la freccia che indica la direzione (per comprenderci in merito a univocità della inversa e molteplicità della diretta); on Nome = Utente indica la condizione da soddisfare per generare correlazione. Nome è un attributo (un campo) di un elemento Utenti, Utente come detto prima è un campo di un elemento della classe MaterialiPrestati.

Automatic sta nel fatto che basterà inserire un elemento in classe MaterialiPrestati per averlo subito correlato con l'elemento padre. Chiaramente l'elemento padre deve esistere nella classe Utenti, altrimenti avviene un fallimento dell'operazione. Come per le associazioni non surgettive, potremo comodamente navigare tra i dati, spostandoci per mezzo delle associazioni tra classi diverse.

Un tipo di associazione non contemplato affatto dai sistemi reticolari è l'associazione con diretta e inversa multipla, come quella di figura 5a tra studenti e esami: molteplicità nel fatto che uno studente ha superato più esami e un esame è stato superato da più studenti. Per descrivere ciò con i meccanismi propri dei sistemi reticolari si ricorre a una classe ausiliaria codominio di due associazioni rispettivamente con le due classi, come mostrato in figura 5 b. Potremmo ad esempio archiviare anche i verbali d'esame in una classe apposita e risolvere il nostro problema. Un verbale d'esa-

me conterrà la matricola dello studente, il codice dell'esame da lui superato, il voto, la data e altre informazioni a scelta. Per navigare nella base, ad esempio per conoscere quali esami ha superato lo studente Rossi, partendo da questo nella classe Studenti scenderemo in Verbali chiedendo al sistema i figli di Rossi in questa classe e per ogni elemento in Verbali chiederemo al sistema i relativi padri in Esami. Analogo discorso per navigazioni in senso opposto, partendo da Esami, volendo conoscere ad esempio quali studenti ne hanno superato uno in particolare.

Facciamo un esempio

In figura 6 è mostrata una ipotetica base di dati per gestire la vendita delle cassette con i programmi pubblicati su MCmicrocomputer. Ipotetica, ripetiamo: restiamo cioè nel didattico-teorico.

Le classi dati usate sono in tutto 5, anche se la classe Utenze sta lì solo per simulare un'associazione con diretta e inversa multipla come nell'esempio di figura 5.

Abbiamo la classe degli Utenti in cui memorizzeremo i vari lettori che hanno ordinato cassette. La classe Programmatori conterrà informazioni circa l'autore del programma; la classe Programmi l'elenco delle varie cassette disponibili e per completezza useremo anche la classe Computer che conterrà l'elenco delle macchine oggi in commercio.

Le associazioni sono tutte surgettive: infatti, ogni programmatore usa un computer, ogni programma gira su un computer, ogni programma è stato scritto da un programmatore e ogni utenza riguarda sia un programma che un utente.

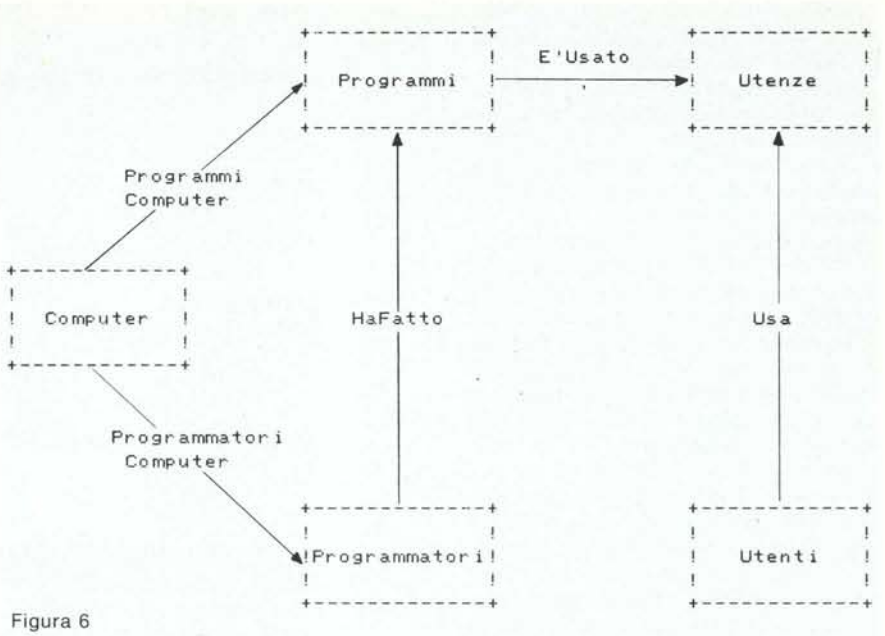

Figura 6

Per prima cosa definiremo la struttura degli elementi delle 5 classi, per semplicità useremo solo campi di tipo stringa.

Computer <-> (Marca, Modello, Memoria)
Programmi <-> (NomeProgramma, Specie, Programmatore, ComputerDaUsare)
Programmatori <-> (Nome, Indirizzo, ComputerUsato)
Utenti <-> (Nome, Recapito)
Utenza <-> (Programma, Utente)

Le associazioni le definiremo così:

ProgrammiComputer automatic set Computer → Programmi on Modello = ComputerDaUsare

ProgrammatoriComputer automatic set Computer →
Programmatori on Modello = ComputerUsato

ProgrammiFatti automatic set Programmatori → Programmi on Programmatore = Nome

È Usato automatic set Programmi → Utenze on Nome = Programma

Usa automatic set Utenti → Utenze on Nome = Utenti

Se al momento attuale volessimo caricare la nostra base di dati dovremmo innan-

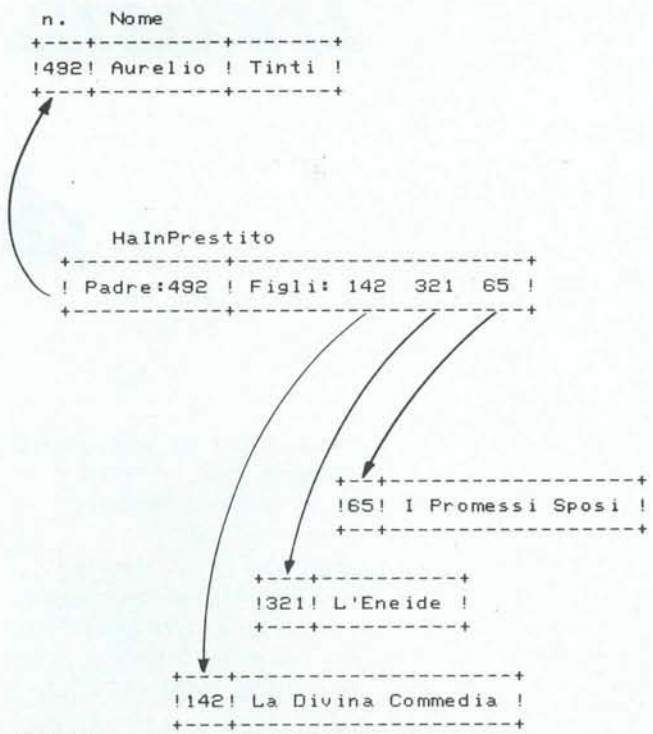

Figura 4

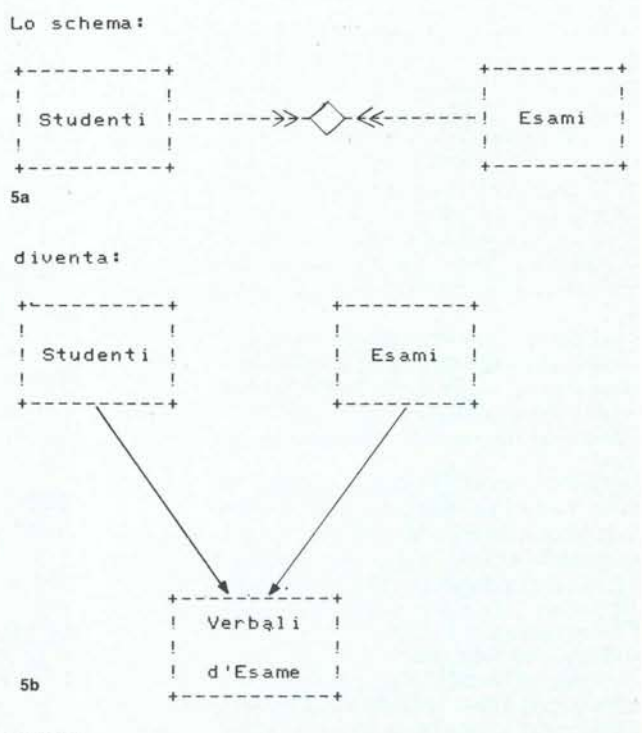

Figura 5

zitutto inserire le varie macchine sul mercato, o almeno quelle di cui sono disponibili programmi su cassetta.

Segue l'inserimento dei vari collaboratori e lettori che hanno contribuito ad ampliare la mini Software Bank di MC. Notare che automaticamente avremo anche correlato i vari programmatori con le macchine della classe computer.

Solo a questo punto potremo inserire l'elenco dei programmi su cassetta, anche loro immediatamente associati alle relative macchine e ai loro padri (programmatori).

Supponiamo ora che il lettore Bartolomeo Cirilli ordini il programma Othello VIC 20, chiaramente già inserito nella base tra le cassette disponibili. Dato che il Cirilli si rivolge per la prima volta a MC, occorrerà inserire prima lo stesso tra gli utenti e poi archiviare l'utenza. Per fare quest'ultima operazione inseriremo nella classe Utenze l'elemento (Othello VIC 20, Bartolomeo Cirilli) che automaticamente si correlerà con i rispettivi elementi nelle classi Programmi e Utenti, venendo a formare di fatto l'associazione tra il Cirilli e il programma acquistato.

Analogamente, se lo stesso richiederà altre cassette, basterà sempre e solo incrementare la classe Utenze con coppie del tipo (<ProgrammaAcquistato>, Bartolomeo Cirilli).

Proviamo ora un po' a farci una bella navigata nella base di dati: ad esempio, si vuol conoscere quale computer ha il lettore Massimo Manzi: sappiamo solo che ha ordinato alcune cassette, ma non ricordiamo quali. La prima operazione da compiere è cercare il Manzi tra gli Utenti, ad esempio con un comando (classico) del tipo:

```
UnUtente = get Utenti with Nome=Massimo Manzi
```

UnUtente è una variabile che ora punta alla registrazione cercata. Potremmo a questo punto chiedere i figli del Manzi in Utenza: ne basterà uno, a noi interessa risalire al computer (si suppone che il Manzi non acquisti cassette per macchine diverse). L'operatore classico per chiedere i figli di un elemento è member; ha come parametro il nome dell'associazione coinvolta essendo possibile che più associazioni colleghino due classi. Scriveremo qualcosa del tipo:

```
UnaUtenza = member (Usa) of UnUtente
```

continuiamo a spostarci, questa volta chiedendo il padre (in Programmi) dell'utenza puntata dalla variabile UnaUtenza:

```
UnProgramma = owner (EUsato) of UnaUtenza
```

l'operatore owner restituisce appunto il padre. La nostra allegra navigata si conclude chiedendo il padre in Computer del programma trovato (che se non avete perso il filo è uno dei programmi del Manzi):

```
IlComputer = owner(ProgrammiComputer) of UnProgramma
```

il tutto si conclude con

```
Print Marca, Modello of IlComputer
```

vedremo cioè apparire marca e modello del computer dell'utente lettore Massimo Manzi. Possiamo ammainare le vele. MC

PIEMONTE
Aba Elettronica
Via Fossati, 5/c
10141 Torino
Tel. 011/332065

Gruppo Sistemi Torino S.r.l.
Strada Torino, 90 h
10092 Beinasco (TO)
Tel. 011/651114

Inter-Rep S.p.A.
Via Orbetello, 98
10148 Torino
Tel. 011/2165901-2165911

LOMBARDIA
Sirius Technology
Via Imperia, 23
20142 Milano
Tel. 02/8467304

Eledra System S.p.A.
Via Ferruccio, 2
20100 Milano
Tel. 02/3492010

VENETO-VENEZIA GIULIA
Corel Italiana S.r.l.
Via Mercatovecchio, 28
33100 Udine
Tel. 0432/291466-480857

Cash S.r.l.
Via Noventa Vicentina, 2
36100 Vicenza
Tel. 0444/507155

Vecomp S.r.l.
Via Chioda, 76
37136 Verona
Tel. 045/583711

Seda s.a.s.
Via Sighele, 7/1
38100 Trento
Tel. 0461/984564

LIGURIA
Siragusa Giuseppe
Via Milano, 85/a
16126 Genova
Tel. 010/261655

EMILIA ROMAGNA
S.H.R. S.r.l.
Casella Postale 275
48100 Ravenna
Tel. 0544/463200

Tempo Reale
Via Centotrecento, 1/A
40126 Bologna
Tel. 051/270701

Maser s.a.s.
Via Corticella, 177
40128 Bologna
Tel. 051/326420

TOSCANA
It-Lab S.r.l.
Via XXIV Maggio, 101
56100 Pisa
Tel. 050/501359

M.T.S. s.a.s.
V.le Guidoni, 93/Z
50100 Firenze
Tel. 055/410996

E.V.M.
Via Marconi, 9/A
52025 Montevarchi (AR)
Tel. 0575/982513