Glauco Spanghero

Introduzione alla progettazione ed alla realizzazione dei database con Microsoft Access®

Trieste, gennaio 2006

Sommario

Indice delle figure

1 Premessa

L'idea di realizzare il presente lavoro è nata sulla scia degli ottimi risultati raggiunti dai miei colleghi e amici Prof. Lucio Torelli e Prof. Massimo Borelli con le loro dispense *"Primi passi in MS Excel®", "Problemi di Statistica Biomedica con MS Excel®"* e *"L'inferenza Statistica: problemi biomedici risolti con il computer"*, edite rispettivamente negli anni 2002, 2003 e 2004 dal Dipartimento di Matematica e Informatica dell'Università degli Studi di Trieste.

Più volte, nel corso di questo periodo di tempo, ci siamo incontrati e abbiamo parlato della possibilità di integrare le tematiche da loro già trattate realizzando una nuova dispensa che introducesse il lettore nel vasto e complesso mondo della progettazione e della realizzazione dei database relazionali, completando quindi le questioni relative *all'elaborazione dei dati* con quelle complementari della *archiviazione e della gestione degli stessi*.

In coerenza con lo spirito da me totalmente condiviso che caratterizza le dispense realizzate dai colleghi, contraddistinte da un baricentro fortemente spostato sulle questioni applicative delle scienze statistiche e informatiche in campo biomedico, il taglio di questo quaderno mantiene la loro stessa struttura concettuale con lo scopo principale di far comprendere al lettore *come* si realizza un database, tralasciando laddove possibile e per quanto (il più) possibile gli aspetti teorici (non per questo poco importanti).

Per questi ultimi ovviamente si rimanda all'immensa letteratura in materia solo parzialmente citata in bibliografia.

Buona parte degli esempi applicativi presentati nascono da un riordino dei miei appunti relativi alle lezioni che ho tenuto in questi anni agli allievi del corso di *"Sistemi di elaborazione delle informazioni"* nell'ambito del Corso di Laurea in Ostetricia e del Corso di Laurea in Infermieristica nella Facoltà di Medicina e Chirurgia dell'Università degli Studi di Trieste.

Per quanto concerne le convenzioni stilistiche, mi sono limitato a riproporre il modello già testato e collaudato dai miei colleghi. Si inizia con un generale

Problema introduttivo
...

che fornisce lo spunto per trattare le problematiche presentate e le relative possibilità di soluzione. Le parole scritte in carattere svizzero (Arial) rappresentano i comandi reperibili nei menu di Microsoft Access®. Alla fine dei capitoli più significativi si trova qualche

Esercizio
...

che il lettore dovrebbe essere in grado di eseguire utilizzando ed adattando gli argomenti che aveva visto fino a quel momento. Di ogni esercizio, al capitolo 6 viene comunque riportata una possibile soluzione. Infine, come di consueto, copie gratuite della presente dispensa si possono scaricare dall'indirizzo:

http://www.dmi.units.it/corsi/index.html

1.1 Sulle banche dati in generale

Prima o poi, in quasi tutte le attività gestionali *si deve affrontare il problema di archiviare dati per poterli successivamente recuperare, consultare ed elaborare*.
Prima dell'avvento dei personal computer, ed in molti casi ancora oggi nonostante la disponibilità di potenti elaboratori e sofisticati software a costi più che accessibili, queste operazioni di archiviazione venivano (e vengono) fatte utilizzando i tradizionali strumenti cartacei ed i relativi archivi fisici costituiti da scaffali, scatoloni, armadi, sottoscala, soffitte e quant'altro in grado di contenere i grossi volumi di materiale prodotti. Non va tuttavia dimenticato che la necessità di continuare con sistemi tradizionali di archiviazione fisica dei documenti in formato cartaceo può derivare anche da motivi di ordine giuridico: si pensi per esempio a pratiche legali, contabili e altre ancora.

Se però si fa una riflessione sulle varie tipologie di dati che normalmente vengono archiviate per essere successivamente utilizzate, ci si accorge che in una grande maggioranza dei casi le informazioni che servono per gestire queste attività non necessariamente devono essere stampate su un pezzo di carta, ma possono essere benissimo rese disponibili mediante la sola visualizzazione su un monitor di personal computer, senza che ciò intacchi minimamente l'efficacia e la qualità del dato di cui si ha bisogno. Inoltre, molto spesso, l'utente che accede alla banca dati non è interessato all'intero documento, ma piuttosto a specifiche informazioni in esso contenute, magari da reperire attraverso i cosiddetti "motori di ricerca" basati sull'inserimento di parole chiave. Infine, e forse questo è l'aspetto più importante, in molti casi è necessario estrarre i dati di un archivio in una forma aggregata, in cui le informazioni siano restituite solamente dopo opportuni processi di elaborazione automatica, cosa che diventa possibile solamente se si dispone di un database informatizzato.
Esempi in tal senso si possono ritrovare nell'elenco, assolutamente non esaustivo, di seguito riportato:

- Server di posta elettronica (e-mail)
- Elenchi telefonici
- Quotidiani "virtuali" su CD o in rete
- SMS ed MMS con telefonia cellulare
- Archivi anagrafici dei Comuni
- Banche dati ambientali e sanitarie
- Gestione dei movimenti bancari
- ...

1.2 Che cos'è un database ?

La seguente definizione di database può tuttora ritenersi valida:

> *"Un database è una collezione di informazioni registrate in formato leggibile dall'elaboratore elettronico e relativa ad un preciso dominio di conoscenze, organizzata allo scopo di poter essere consultata dai suoi utilizzatori".*

Il problema di progettare un database può essere conseguentemente formulato come segue:

> *"Progettare la struttura logica e fisica di una o più basi di dati al fine di gestire le esigenze informative degli utenti di un'organizzazione per un insieme ben definito di applicazioni".*

Conseguentemente, un Data Base Management System (DBMS) è un software in grado di gestire un database per rappresentare le informazioni di interesse in esso contenute.
Un DBMS si occupa quindi della gestione informatica di un database e in questa dispensa verrà utilizzato Microsoft Access®.
I database devono possedere almeno le seguenti caratteristiche:

- **Grandi**: nel senso che possono avere anche dimensioni enormi (terabyte[1] e oltre).
- **Condivisi**: perché deve essere possibile a più utenti l'accesso contemporaneo ai dati comuni contenuti nella memoria centrale di una rete.
- **Affidabili**: deve essere garantita l'integrità dei dati, anche in caso di malfunzionamento hardware e/o software, prevedendo procedure di backup e, in caso di crash, di recupero dati relativamente almeno all'ultima delle transazioni di aggiornamento.
- **Privacy**: i dati devono poter essere consultati prevedendo, all'occorrenza, specifiche autorizzazioni di accesso, alle quali devono essere associati i *diritti* di ciascun utente (sola lettura, lettura e scrittura, modifica della struttura, ecc.).

Per chiudere questo capitolo introduttivo, diremo solo che la soluzione del problema *"come si fa un database"* si ottiene risolvendo due questioni complementari: da un lato bisognerà *"saper progettare il database"* e dall'altro bisognerà *"saper scegliere ed utilizzare lo strumento informatico per realizzare fisicamente e gestire il database"*.
A scanso di equivoci, avvertiamo che la prima parte della presente dispensa, dedicata esclusivamente alla progettazione dei database, non conterrà alcun riferimento né a Microsoft Access® né più in generale all'informatica per quella che è l'accezione più familiare del termine.
Quindi per affrontarla efficacemente basterà disporre di carta, penna e, ovviamente, buona volontà. Lasciate quindi spento il computer ed accendete il cervello !

[1] 1 TB = 1 TeraByte = 10^{12} Byte

1.3 Prerequisiti (sottotitolo: Access è facile)

Al livello a cui vogliamo arrivare, che non è assolutamente elementare, per progettare un database è sufficiente saper leggere, scrivere e ragionare. Per utilizzare Microsoft Access® sarà sufficiente possedere le semplici nozioni sull'uso di un computer in ambiente Microsoft Windows®. Pare proprio che non occorra poi molto.

2 La modellistica

2.1 Per iniziare

Per cercare di rendere più chiaro possibile il percorso che porta alla realizzazione di un database, d'ora in poi faremo riferimento ad un unico *caso studio* da utilizzare come base di lavoro, denominato *"registrazione visite mediche"*.

Caso studio "registrazione visite mediche"
Un vostro collega, che sta gestendo il registro delle prestazioni sanitarie nell'ospedale dove lavorate e dove i pazienti accedono periodicamente per effettuare le visite mediche, vi chiede un consiglio su come realizzare in un archivio degli accessi dei pazienti stessi.
Come pensate di aiutarlo ?

Ogni volta che ci si trova di fronte alla necessità di ricercare una soluzione ad un problema connesso con una realtà complessa della vita reale, costituito da una moltitudine di variabili spesso imprevedibili o di difficile misurazione, è buona norma ridurre questa realtà complessa ad un *modello semplificato*, costituito a sua volta da una serie di componenti elementari facilmente controllabili e gestibili.

Quando un ingegnere deve progettare una struttura in cemento armato affinché possa resistere alle sollecitazioni richieste, oppure quando un geologo deve progettare un pozzo per il prelievo di una certa portata di acqua dal sottosuolo, oppure ancora quando un metereologo deve prevedere il tempo che farà domani, in tutti questi casi la base per i loro calcoli non è certamente costituita dalla complicata realtà della natura tal quale, bensì da modelli più o meno semplificati che consentono di ottenere soluzioni approssimate con un livello di indeterminatezza (errore) accettabile in funzione degli obiettivi prefissati[2].

Quindi, per ritornare al problema introduttivo, la ricerca della soluzione non può prescindere dal trovare una risposta chiara alla seguente domanda: cosa intendiamo per *"registrazione degli accessi dei pazienti" ?*

Le risposte possibili possono essere molte, e tutte sostanzialmente e formalmente corrette. Tra tutte bisognerà saper scegliere quella che saprà fornirci il maggior numero di informazioni minimizzando il dispendio di risorse e le eventuali imprecisioni.

Ecco quindi che dovremo lavorare per costruire nella nostra mente un *modello approssimato*, sostanzialmente diverso dalla realtà, ma ad essa sufficientemente simile al punto di essere capace di contenere comunque tutte le informazioni di cui abbiamo bisogno nel contesto in questione. Con una particolarità in più: avendolo costruito noi ad immagine della realtà, useremo l'accortezza di fare in modo che tutti i suoi elementi siano perfettamente controllabili.

2.2 La raccolta e l'analisi dei requisiti

La progettazione di una base dati presuppone una chiara e dettagliata conoscenza *a priori* delle aspettative degli utenti destinatari dello strumento che si intende realizzare.

Tutte le attività propedeutiche alla fase progettuale vengono sinteticamente denominate *raccolta ed analisi dei requisiti.* Queste, almeno per quanto concerne i grossi sistemi, vengono svolte da un'equipe di analisti che spesso vedono il coinvolgimento anche del cliente stesso o comunque del destinatario finale del prodotto.

Per questa fase del lavoro, assimilabile in tutto e per tutto agli incontri preliminari che, ad esempio, un'agenzia immobiliare programma con la clientela per tastare il polso ed individuare con maggiore chiarezza possibile quelli che sono gli stili preferiti dal

[2] Normalmente, in modellistica, si tende a realizzare (e scegliere) i cosiddetti *modelli conservativi*, ovvero quelle rappresentazioni semplificate che consentono di studiare la realtà e di prevedere i suoi comportamenti *nella peggiore delle ipotesi possibili*. Quindi se il modello "commette errori", questi saranno sempre orientati verso una sovrastima degli effetti (le travi dell'ingegnere sopporteranno *più peso* di quello dichiarato, il pozzo del geologo consentirà di emungere *più acqua* di quella effettivamente prelevata, e via dicendo).

potenziale acquirente della casa, si ritiene opportuno consigliare vivamente un approccio di gruppo con il coinvolgimento di tutte le parti interessate (*stakeholder*). Eventuali errori in fase di analisi dei requisiti comportano spesso l'insoddisfazione dell'utente finale, il quale potrebbe non ritrovare nel prodotto implementato i requisiti e le caratteristiche necessari a quelli che erano i suoi fini pratici, con la conseguenza purtroppo abbastanza frequente di non utilizzare affatto il sistema. Sostanzialmente la realizzazione di un database, una volta effettuata l'analisi dei requisiti, passa attraverso le seguenti tre fasi distinte:

- Costruzione del modello concettuale (capitolo 2.3)
- Derivazione del modello logico (capitolo 2.4)
- Implementazione del modello fisico (capitolo 3)

2.3 Il modello concettuale Entity/Relationship (E/R)


Peter P. Chen

La realizzazione del "modello approssimato", termine che ora abbandoneremo per utilizzare un meno pittoresco ma formalmente più corretto *"modello concettuale"*, obbliga il modellista a scegliere un metodo condiviso di rappresentazione del *caso studio* per quanto concerne sia la *forma grafica* sia la *sostanza della rappresentazione* che si appresta a costruire. Ovvio che qui non inventeremo nulla di nuovo, bensì adotteremo uno standard di riferimento denominato *Entity/Relationship*[3] *(E/R)* introdotto per la prima volta da P. Chen[4] nel 1976. Per farla breve, anche in coerenza con quanto abbiamo detto in premessa, il modello concettuale E/R rappresenta un'astrazione di alto livello[5] (e di bassa

[3] *"Entity-Relationship"* va tradotto correttamente in *"Entità-Associazione"*, anche per non confondere le idee con quello che invece è il Modello Relazionale, meno astratto, che vedremo più avanti nel capitolo 2.4.
[4] Chen, P.P.: *"The Entity Relationship Mode - Torward a Unified View of Data"*, TODS, Vol. 1, Numero 1, (03/1976).
[5] Di norma, più alto è il livello di astrazione con cui si vuole rappresentare un caso studio e più basso sarà il corrispondente grado di complessità. Si immagini per esempio di voler rappresentare il caso studio *"viaggio in automobile"*; in maniera molto astratta esso potrà benissimo essere rappresentato dalle entità automobile, guidatore, strada, località (di arrivo e di partenza). Bisognerà poi analizzare le eventuali *associazioni* tra queste entità: il guidatore *guida* l'automobile, l'automobile *percorre una strada*, la strada *collega due località*. Se poi, in una fase successiva, si vorrà abbassare il livello di astrazione per aumentare la complessità del modello, bisognerà definire più precisamente cosa si intende (nel modello) per automobile, guidatore, strada e località, stabilendo i cosiddetti *attributi delle singole entità*. Infine si potranno analizzare le eventuali *relazioni* tra queste entità *(sì, è scritto giusto, relazioni e non semplicemente associazioni)*: *quale guidatore guida* l'automobile, *quanto l'automobile percorre una strada*, *come la strada collega due località* e via dicendo. In questo modo si passerà dal cosiddetto *modello concettuale* al più dettagliato *modello logico*. Ma di ciò parleremo più avanti.

complessità) che consente di individuare, rappresentandole, le cosiddette *entità* del caso studio e, tra esse, di stabilire se e quali siano le relative *associazioni*.

2.3.1 Gli "attori" del modello concettuale E/R

Come già accennato, nella fase di realizzazione del modello concettuale è fondamentale saper individuare le *entità* del sistema e, tra esse, le (se ci sono) *associazioni*.
Considerati gli scopi della presente dispensa, la semantica dei concetti di *entità ed associazione* va ricercata in quella che è la loro comune interpretazione quotidiana.
Entità saranno tutti quegli "oggetti" dotati di esistenza fisica nella realtà descritta dal caso studio, come per esempio i pazienti, i reparti, ecc., oppure anche "oggetti" esistenti a livello concettuale come le visite mediche.
Nel caso studio proposto, saranno sicuramente *entità* i seguenti "oggetti":

- pazienti (realtà fisica)
- reparti (realtà fisica)
- personale sanitario (realtà fisica)
- visite mediche (realtà concettuale)

Associazioni saranno invece tutti gli (eventuali) *legami* che riterremo esistenti tra le varie entità individuate. Esempi in tal senso potranno essere:

- il paziente *accede* al reparto
- il medico *effettua* la visita

Il passo più difficile, partendo dal caso studio proposto all'inizio del capitolo 2.1, è appunto quello di eseguire l'analisi dei requisiti per l'individuazione in esso delle entità e delle associazioni più efficaci e rappresentative.
Una prima regola, che come tale ha molte eccezioni, consiste nell'effettuare un'analisi logica della frase individuando i *sostantivi* ed i *verbi (anche sottintesi).* Nei primi potremmo iniziare a cercare le *entità*, mentre nei secondi le *associazioni*. Il condizionale è d'obbligo.

2.3.2 Simbologie grafiche e convenzioni stilistiche

Molto semplicemente, tralasciando alcuni aspetti più approfonditi, il modello concettuale E/R propone una serie di convenzioni stilistiche che consentono di "disegnare" il modello stesso con un certo rigore sia della forma sia della sostanza, come rappresentato in Figura 1. Ogni entità deve essere a sua volta descritta tramite i cosiddetti *attributi* (vedi § 2.3.4), che rappresentano quelle proprietà utili a descrivere più compiutamente l'entità.

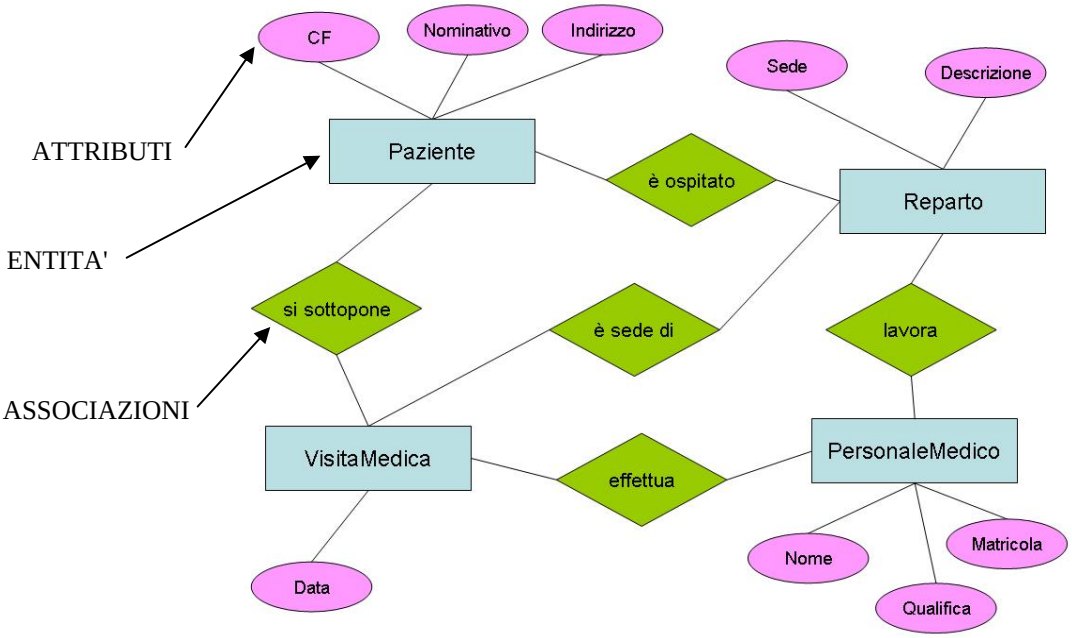


Figura 1: Convenzioni stilistiche e grafiche della rappresentazione del modello concettuale E/R relativamente al caso studio proposto: i rettangoli rappresentano le entità (descritte con il nome in essi contenuto), le linee con i rombi rappresentano le associazioni e gli ovali descrivono gli attributi.

2.3.3 Una piccola regola, semplice e molto utile (soprattutto in futuro)

E' buona norma scrivere i nomi delle entità, delle associazioni e degli attributi senza usare spazi vuoti.

In questa prima fase di modellistica, questa regola, anche se non rispettata, non comporta particolari conseguenze negative. Ma successivamente, soprattutto quando si inizierà la scrittura del database con Microsoft Access® e si esplorerà il linguaggio di interrogazione SQL, la presenza di spazi vuoti nei nomi potrà causare notevoli problemi interpretativi da parte del DBMS.

Quindi, anche perché non costa proprio niente, è bene abituarsi fin da subito a scrivere le parole composte che identificano le entità avendo cura di non mettere spazi vuoti, ovvero "riempirli" per esempio con il simbolo comunemente usato _ (si chiama underscore e si trova nella parte bassa a destra della tastiera).

Quindi VANNO BENE i nomi VisitaMedica, Visita_Medica, visitamedica (brutto!), mentre NON VANNO BENE ad esempio i nomi Visita medica, Visita.Medica (mai e poi mai usare il "punto" per separare parole; il punto ha un preciso significato nei formati numerici e/o come separatore per indicare l'attributo di un file, quindi c'è un rischio altissimo che il suo utilizzo comporti seri problemi di funzionamento).

2.3.4 Gli attributi

Ogni entità individuata nel modello concettuale E/R deve essere compiutamente descritta mediante una serie di proprietà denominate *attributi*.
Anche se ancora non ci siamo arrivati, per maggiore chiarezza anticipiamo che l'evoluzione del modello concettuale ad un livello di astrazione più basso, denominato *modello logico*, vede le entità "trasformarsi" in relazioni, ovvero tabelle (vedremo meglio in seguito l'equivalenza di questi due termini).
Gli attributi individuati nel modello concettuale diventano i campi delle relazioni, *ovvero le intestazioni delle colonne delle tabelle contenenti i dati.*
Gli attributi si dividono in:

- Attributi semplici
- Attributi composti
- Attributi derivati (o calcolati)

Gli *attributi semplici* (o atomici) sono quelli che non possono essere scomposti in componenti più elementari. A questa categoria appartiene per esempio la Matricola del PersonaleMedico.
Gli *attributi composti*, come implicito dal nome, sono quelli che, almeno teoricamente (non è detto che serva o che convenga farlo) possono essere scomposti in più attributi semplici. A questa categoria appartiene per esempio il Nominativo del Paziente (nel Nominativo infatti ci si prepara ad inserire la stringa Mario Rossi, costituita dal Nome Mario e dal Cognome Rossi. L'attributo Nominativo può quindi essere scomposto in due attributi semplici Nome e Cognome).
Gli *attributi derivati*, più comuni di quanto sembra, possono essere ottenuti mediante "calcoli", nell'accezione più ampia del termine, sugli attributi semplici e/o composti esistenti. L'attributo Età ne è un classico esempio, in quanto può essere ottenuto dalla differenza tra la "data di oggi" e la "data di nascita". L'attributo DataNascita è l'attributo

semplice, mentre la data di oggi viene calcolata automaticamente mediante un'apposita funzione presente in ogni DBMS. Un altro esempio di attributo derivato è il Codice Fiscale, ottenibile mediante un algoritmo a partire dal Nome, Cognome, Data di nascita, Comune di Nascita e Sesso (tutti attributi semplici). Per quanto concerne gli attributi derivati la questione non è solo formale, bensì nasconde un aspetto sostanziale denominato *ridondanza*. Prevedere nel modello una serie di attributi semplici e anche un attributo derivato da questi attributi semplici corrisponde a generare una ripetizione (più correttamente ridondanza), col rischio di commettere errori, come per esempio sbagliare la registrazione di un'età dopo che si è inserita correttamente la data di nascita: quale delle due è l'informazioni è quella giusta se non mi accorgo dell'errore ?

2.3.5 La modellistica è un processo iterativo

Il titolo di questo paragrafo è già di per sé un'affermazione.

Arrivati fino qua ci si è resi sicuramente conto che i contenuti di un modello concettuale non sono immediatamente chiari all'analista, che spesso deve immedesimarsi in chi propone il caso studio per cercare di capire le intenzioni e i desideri di quest'ultimo.

Si tratta in sostanza di "entrare nella testa" di chi ha posto il caso studio, traducendone la prosa che lo descrive in una serie di "oggetti" astratti da tradurre in una struttura concettuale, logica e fisica gestibile con un DBMS.

Tutto ciò premesso, sarebbe quantomeno presuntuoso pensare di arrivare ad uno schema perfetto dopo solamente un unico tentativo.

E' doveroso sottolineare, a rischio di ripeterci, che l'evento "registrazione degli accessi dei pazienti", in realtà, si compone di un numero enorme di sfaccettature.

C'è il paziente, con la sua anamnesi, il suo carattere, le sue caratteristiche somatiche e, perché no, anche il suo colore di capelli. Egli arriva all'ospedale, parcheggia la macchina, magari litiga con qualche altro automobilista, poi sale nel reparto, suona il campanello ed entra. Lo accoglie una gentile infermiera che lo fa accomodare in sala d'aspetto, dove egli sfoglia una rivista per ingannare il tempo. Quando arriva il suo turno si alza, entra nell'ambulatorio ed incontra il medico, che inizia ponendogli qualche domanda di rito prima di sottoporlo alla visita medica vera e propria, anch'essa caratterizzata da una moltitudine di variabili distinte.

Infine il paziente soddisfatto (si spera), saluta, esce e se ne va.

Ebbene, tutta questa serie di eventi, molti dei quali senza soluzione di continuità, *noi abbiamo deciso di sintetizzarli in uno schema concettuale semplice come quello rappresentato in* Figura 1.

La questione è tutta qui: la serie di entità/associazioni che abbiamo scelto modellizza in modo completo ed esaustivo l'evento "registrazione degli accessi dei pazienti"? Essa è sufficiente a rispondere a tutte le domande che immediatamente, oppure in futuro, vorremo porci in relazione alla serie di registrazioni effettuata? Abbiamo mantenuto gli attributi necessari e scartato quelli superflui?

Detto questo mettiamoci pure il cuore in pace, affermando che il modello concettuale *ottimale* si ottiene solamente con molta esperienza e dopo una serie di tentativi via via più precisi, spesso mediante un lavoro di gruppo dove ognuno cerca di aggiungere un tassello al puzzle finale. Trattasi di un classico approccio iterativo, che ad ogni nuovo ciclo affina il risultato raggiunto e condiviso nel ciclo precedente. Considerato che la qualità finale del database dipende in tutto e per tutto dalla struttura del suo modello concettuale, gli eventuali errori e le imprecisioni iniziali non potranno che causare un effetto domino su tutto il percorso successivo, causando notevolissimi problemi e perdite di tempo.

Il saggio cinese disse: misura due volte, ma taglia una volta sola. E di lui naturalmente ci fidiamo.

2.3.6 Esercizi

Esercizio 1 (facile)
In un reparto di pediatria si desidera realizzare un database per la registrazione dei parti, che possa contenere informazioni anagrafiche sia sui neonati sia sulle madri.

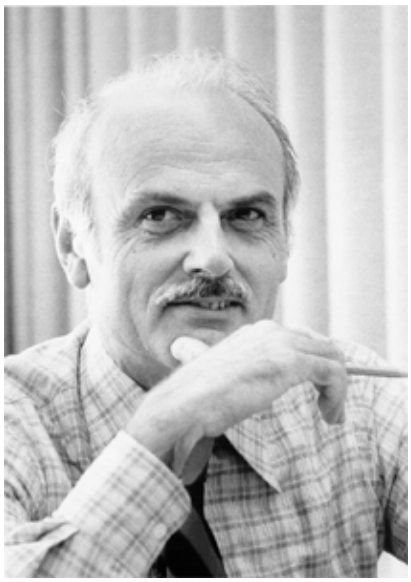
Esercizio 2 (intermedio)
Siccome il disordine in casa ha raggiunto livelli tossici, dovete realizzare un database per archiviare i libri della vostra biblioteca, così da poter ordinarli un po' meglio. Visto che siete degli accaniti lettori, possedete libri di vario genere, di varie case editrici e, per alcuni, capita spesso che li leggiate più volte. Quindi il vostro database dovrebbe contenere anche le registrazioni delle letture fatte.

Esercizio 3 (impegnativo)
In un'azienda, il responsabile della manutenzione si accorge che troppo spesso non vengono rispettate le scadenze per la manutenzione degli autoveicoli in dotazione.
Richiede quindi un database in cui registrare le varie scadenze, passate e future, precisando nel contempo che alcune di esse sono prevedibili nel tempo (p.es. assicurazione, collaudo), mentre altre dipendono dal chilometraggio (p.es. cambio olio, tagliando).

2.4 Il modello logico relazionale di Codd

Un elaboratore elettronico è sostanzialmente una macchina che, opportunamente programmata, è in grado di eseguire con velocità enorme operazioni matematiche sui "numeri" contenuti nella sua memoria. Volutamente il termine "numeri" è racchiuso tra virgolette, in quanto anche le lettere dell'alfabeto, con opportune codifiche, possono essere rappresentate mediante un codice numerico[6]. Quindi l'elaboratore esegue con facilità anche operazioni "matematiche" con lettere o stringhe alfanumeriche. Per lo stesso motivo di prima anche il termine "matematiche" viene racchiuso tra virgolette, in quanto nel contesto in questione l'accezione del termine va estesa anche ad operazioni sulle stringhe alfanumeriche, forse meno note al grande pubblico.

Le basi teoriche delle operazioni matematiche a cui si fa riferimento sono state rigorosamente definite nel 1970 dal matematico inglese Edgar "Ted" Codd del Centro Ricerche IBM[7]. La materia da cui prese spunto prende il nome di *algebra relazionale* che noi, considerato il taglio pratico della presente dispensa, tratteremo in maniera estremamente semplificata per quel tanto che basta a farci proseguire senza intoppi nella realizzazione del nostro database con Microsoft Access®. Nel 1985 Codd pubblicò un documento[8], diventato una specie di pietra miliare per gli sviluppatori, contenente la famosa "lista delle 12 regole" che definiscono il database relazionale ideale.



Edgar "Ted" Codd
(1923 - 2003)

2.4.1 Ci sono limiti nel modello concettuale E/R ?

L'elevato grado di astrazione del modello E/R, nel quale ad esempio non sono definite le regole di calcolo e, soprattutto, non viene fatto alcun riferimento al DBMS che sarà utilizzato per le elaborazioni, rende impossibile una sua diretta applicazione informatica.

[6] Per i più curiosi si rimanda all'indirizzo http://www.lookuptables.com/
Esperimento: dopo aver attivato la tastiera numerica del vostro PC (si deve premere il tasto Bloc Num), aprite un qualsiasi editor di testo (p.es. MS Word), posizionatevi sul cursore e, con il tasto Alt premuto, digitate sulla tastiera numerica il numero 0100 e rilasciate il tasto Alt. Nel documento dovrebbe comparire la lettera d. Il numero 0100 è il cosiddetto *codice ASCII* della lettera d. Per esercizio provate a digitare 0126, 0200, 0230, ...

[7] Codd, E.F.: *"A Relational Model for Large Shared Data Banks"*, CACM, Volume 13, Numero 6, giugno 1970.

[8] Si veda http://en.wikipedia.org/wiki/Codd%27s_12_rules

Anche senza conoscere nulla dell'algebra relazionale, appare abbastanza intuitivo che un diagramma E/R costituito da quadrati, rombi e linee di vario genere, tra l'altro disegnate sulla carta, non può essere interpretato direttamente da un software come un insieme di operazioni da compiere sugli oggetti del diagramma stesso. Occorre tradurre questo modello E/R in un qualcosa di meno astratto (e purtroppo di più complesso) che risulti comprensibile alla macchina ed al software chiamati a fare i calcoli.
Ebbene, il passaggio in questione prevede la cosiddetta *derivazione (dal modello concettuale E/R) del modello logico relazionale (di Codd)*, a sua volta caratterizzato da specifiche regole matematiche che il DBMS sarà in grado di interpretare eseguendo correttamente tutte le operazioni richieste.

2.4.2 Le relazioni (tabelle) e le tuple

In nota 3 al capitolo 2.3 avevamo sottolineato che la traduzione di Entity/Relationship deve essere fatta in Entità/Associazioni (e non Relazioni!), proprio per non confondere le idee con il termine "relazioni" trattate dal modello logico relazionale di Codd.
E' necessario quindi definire il concetto di *relazione* usato nel contesto in esame.
Crediamo che la cosa migliore sia citare testualmente Elmasri e Navathe[9]:

[...] informalmente una relazione assomiglia a una tabella di valori [...] quando si pensa a una relazione come a una tabella di valori, ogni riga della tabella rappresenta una collezione di valori di dati collegati [...] nel modello relazionale ogni riga della tabella (relazione) rappresenta un fatto che generalmente corrisponde a un'entità o a un'associazione del mondo reale (già individuate nel modello E/R).

Finalmente ecco che, con il passaggio al modello relazionale, i concetti astratti di *Entità* e di *Associazione* assumono i connotati più concreti e palpabili di *Tabella* (ovvero Relazione). Nel proseguo del testo, considerato che i "benefici" derivanti dall'utilizzo del termine *tabella* sono sicuramente prevalenti rispetto i "costi" legati al non utilizzo del quasi-sinonimo *relazione*, utilizzeremo sempre il primo al posto del secondo.
La *tabella* sarà quindi la "traduzione matematica" dell'entità già individuata nel modello concettuale, e gli attributi saranno i *campi* della tabella stessa (ovvero l'intestazione delle

[9] Elmasri, R.A., Navathe, S.B.: *"Sistemi di basi di dati. Fondamenti."* IV Edizione, Ed. Pearson/Addison Wesley, settembre 2004.

sue colonne). La tabella avrà quindi tante colonne quanti saranno gli attributi dell'entità dalla quale deriva.

Ogni riga della tabella rappresenta una collezione di valori di dati collegati tra loro, denominati *tuple*, ed ogni tupla potrà essere quindi interpretata come un *fatto* del mondo reale, che nel modello logico viene chiamato *istanza* (Figura 2).

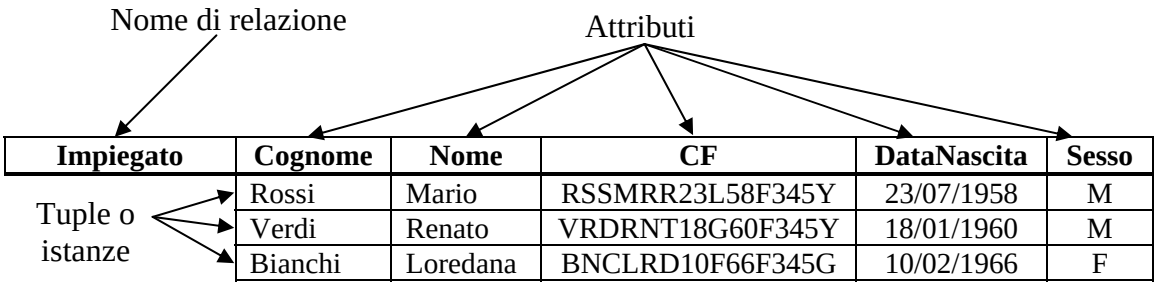


Figura 2: Gli attributi e le tuple di una relazione esemplificativa IMPIEGATO con la simbologia convenzionale di Codd.

2.4.3 Le chiavi

Uno dei presupposti del modello relazionale è rappresentato dal fatto che ogni istanza (ovvero tupla) non deve essere interpretata con ambiguità. Deve essere possibile cioè individuare con assoluta certezza ognuna delle istanze nella tabella, il che equivale ad ammettere che in una stessa tabella non devono esserci istanze duplicate, ovvero due istanze che presentano *tutti* i loro valori collegati uguali. Sarà quindi sufficiente che *almeno uno dei valori dell'istanza non possa essere duplicato*. Infatti nel modello relazionale ogni tabella è un insieme.

Nell'esempio di Figura 2 il CF, stante proprio la sua definizione giuridica, è sicuramente un attributo che non ammette valori duplicati, essendo diverso da persona a persona.

Un siffatto attributo viene definito *chiave*. Ogni tabella deve avere almeno una chiave e tra tutte le possibili chiavi, denominate appunto *chiavi candidate*, una a scelta sarà definita *chiave primaria* (PK = Primary Key nella notazione comune). Il problema della scelta di una PK nell'insieme delle chiavi candidate è un'operazione a discrezione del modellista, anche se spesso è difficile individuare più di un attributo con caratteristiche di chiave, quindi di fatto la scelta ricade giocoforza sull'unico di essi a possedere il requisito di non duplicabilità.

2.4.4 Le tabelle del caso studio "registrazione visite mediche"

Tornando al modello E/R di Figura 1 relativo al caso studio in esame, si può osservare che la base dati contiene almeno quattro tabelle, individuabili in prima approssimazione nelle quattro entità *Paziente*, *VisitaMedica*, *Reparto e PersonaleMedico*. Volutamente si è parlato di *almeno quattro tabelle*, in quanto anche la derivazione del modello relazionale è un processo iterativo e spesso, a mano a mano che si prosegue nella sua realizzazione, "emergono" tabelle nuove che magari, nel ciclo precedente, erano "mimetizzate" come attributi dentro una o più delle tabelle esistenti.
Proviamo allora a costruire le tabelle (che "riusciamo a vedere") e, per maggiore chiarezza, in ognuna di esse inseriamo qualche istanza esemplificativa.

Paziente	CF	Nominativo	Indirizzo
	RSSMRR30B46F345G	Rossi Mario	Via Dante 12, Trieste
	BNCVVN23G66F344Y	Bianchi Viviana	Via Leopardi 34, Udine
	GLLCRM01L45F443Y	Gialli Carmela	Via Manzoni 1, Pordenone

VisitaMedica	Data	CodProg
	12/03/2005	CHR12032005
	12/03/2005	OTR12032005
	20/06/2005	ORT20062005

PersonaleMedico	Matricola	Nome	Qualifica
	123D456	Verdi Gianni	Ortopedico
	234G567	Rossi Paolo	Pediatra
	136H654	Celeste Aida	Otorinolaringoiatra

Reparto	CodRep	Sede	Descrizione
	OTR01	Ospedale Maggiore - Padiglione A	U.O. Otorinolaringoiatria
	PDT04	Clinica universitaria	Cattedra di Pediatra
	ORT4	Ospedale Maggiore - Padiglione F	U.O. Ortopedia

Figura 3: Le tabelle derivate dalla traduzione del modello E/R di Figura 1. Si osservi che, in alcune di esse, sono stati introdotti ex novo i campi chiave primaria (PK), evidenziati in sottolineato, che non erano individuati negli attributi del modello concettuale, perché non necessari vista l'astrazione del modello stesso.

2.4.5 Dalle singole tabelle allo schema di base dati: le chiavi esterne

Una volta realizzate tutte le tabelle ed in base alle associazioni precedentemente individuate nel modello E/R, diventa possibile costruire il cosiddetto *schema di base dati relazionale*.

Il problema questo: le sole tabelle costruite in Figura 3, come si può facilmente osservare, non consentono di rappresentare le informazioni implicite delle associazioni del modello E/R. Cerchiamo di spiegarci meglio con un esempio.

Prendiamo la tabella VisitaMedica ed analizziamo le istanze esemplificative in essa registrate. Appare abbastanza chiaro che in esse non c'è nessun attributo che *leghi le istanze visite mediche ai pazienti* ad esse sottoposti. Nelle tuple troviamo infatti *solo* informazioni relative alle visite mediche e nient'altro. Parimenti nelle stesse tuple non c'è alcuna informazione che stabilisca legami con i *medici* che le fanno né con i *reparti ospedalieri* afferenti. Ovviamente, con le opportune diversificazioni, lo stesso discorso vale anche per tutte le altre tabelle dello schema.

La soluzione a questo problema è semplice: in base alle associazioni individuate nel modello E/R, basterà aggiungere in ogni tabella del modello logico di Figura 3 l'attributo PK delle tabelle ad esse collegate che, nella loro veste di elemento caratterizzante del collegamento, assumeranno il nome di *chiavi esterne (FK = Foreign Key).* In Figura 4 è rappresentato lo schema di base dati relazionale integrato con gli attributi FK che collegano tra loro le tabelle del modello (sono omesse le tuple, mentre sono evidenziate le associazioni).

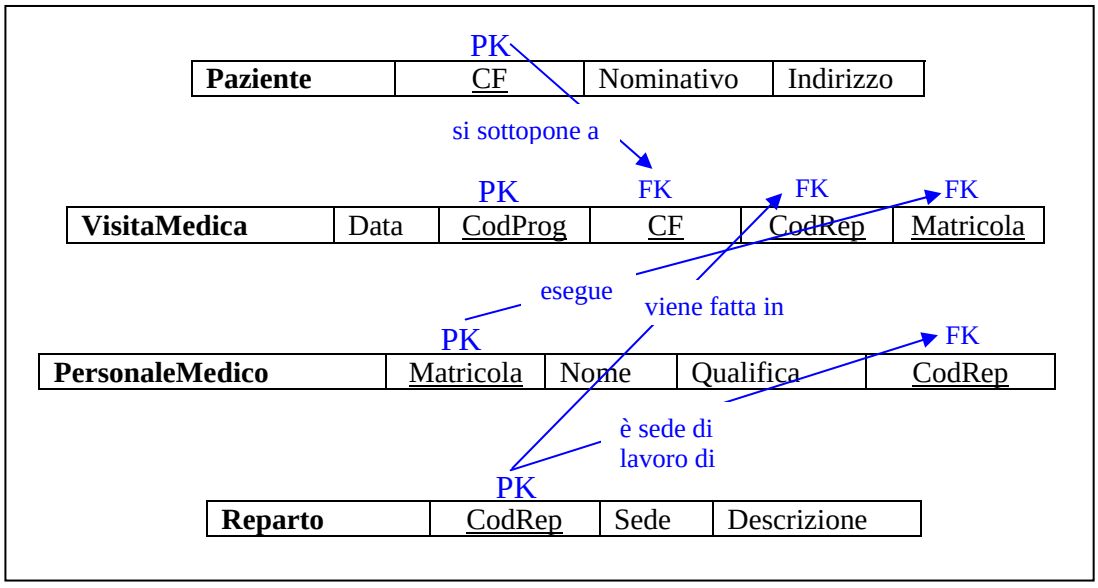


Figura 4: Schema di base dati relazionale con inserimento delle chiavi esterne (FK) nelle tabelle tra loro associate. Per chiarezza sono esplicitati i *verbi* delle associazioni.

Sarà il DBMS che, sulla base dello schema di base dati, provvederà a gestire le istanze delle tabelle associate. Per vedere meglio il risultato pratico dello schema, nell'esempio di Figura 5 sono esplicitate due istanze in VisitaMedica.

VisitaMedica

Data	CodProg	CF	CodRep	Matricola
20/06/2005	ORT20062005	BNCVVN23G66F344Y	ORT4	123D456
12/03/2005	OTR12032005	GLLCRM01L45F443Y	OTR01	136H654

Figura 5: Esempio di istanze nella tabella VisitaMedica.

2.4.6 La cardinalità delle associazioni binarie tra tabelle

Nel capitolo precedente abbiamo potuto osservare che il modo per definire matematicamente *l'associazione (binaria)* tra tabelle è quello di inserire gli attributi FK. Per maggiore chiarezza e completezza formale, oltre che per la sua estrema importanza nell'ambito del modello logico relazionale, risulta tuttavia necessario procedere ad un approfondimento sul tema delle associazioni binarie.

Sinteticamente, in un modello relazionale le associazioni binarie tra tabelle collegate possono essere di tre tipi:

- Associazione binaria 1:1 (si legge "uno ad uno")
- Associazione binaria 1:N (è scritta anche nella forma $1 \rightarrow \infty$ e si legge "uno a molti")
- Associazione binaria N:M (è scritta anche nella forma $\infty \rightarrow \infty$ e si legge "molti a molti")

Questi tipi di associazione vengono chiamati *cardinalità* e nello schema di base dati relazionale devono essere esplicitamente definiti, in quanto essi rappresentano un elemento portante di tutta la struttura logica del modello, al pari delle tabelle e dei loro relativi attributi. Approfittiamo inoltre del momento quantomai opportuno per passare ad una nuova rappresentazione grafica delle tabelle, concettualmente identica a quella già utilizzata nei capitoli precedenti ma formalmente diversa e più simile al modo familiare con cui Microsoft Access® ci consentirà di rappresentare questi oggetti nelle sue interfacce di progettazione.

Vediamo quindi come si *risolvono le cardinalità*.

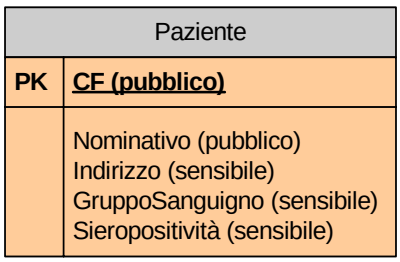
CASO 1:1

Esempio

Si pensi ad una tabella rappresentativa di un'entità *Paziente*. Tutti sanno che, per motivi di privacy, è bene che le informazioni riguardanti alcuni attributi sensibili di una qualsiasi persona in generale, e di un paziente in particolare, siano mantenute "nascoste", ovvero

siano registrate in modo da agevolare il loro mascheramento in caso di accesso alla banca dati. Senza entrare troppo nei dettagli sui motivi pratici che inducono la scelta che stiamo per fare, si sappia che in tali circostanze è conveniente dividere la tabella *Paziente* in due tabelle complementari, che potremmo chiamare *AnagraficaPaziente* e *PrivacyPaziente*.
Nella prima inseriremo tutti e soli gli attributi "pubblicabili", mentre nella seconda metteremo quelli "sensibili".
La PK di *Paziente*, rappresentata da CF, dovrà essere inserita sia come PK in *AnagraficaPaziente* che come PK in *PrivacyPaziente*. In questo modo ogni istanza di *AnagraficaPaziente* "vedrà" una ed una sola istanza di *PrivacyPaziente* e viceversa, andando perciò a costituire un'associazione binaria 1:1 correttamente gestibile dal DBMS.

... un'unica tabella Paziente ...



... si trasforma in due tabelle + un'associazione 1:1 ...

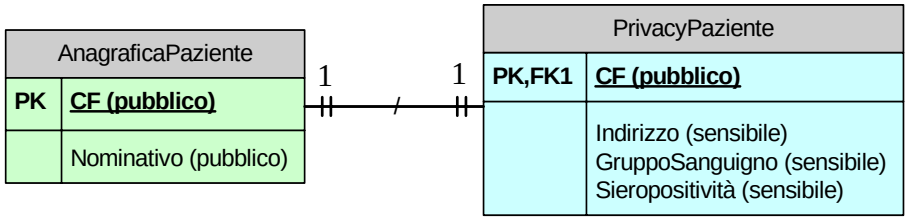


Figura 6: Esempio di associazione 1:1. La tabella *Paziente* viene sostituita da due tabelle *AnagraficaPaziente* e *PrivacyPaziente* nelle quali la PK è sempre la stessa della tabella di partenza.

CASO 1:N

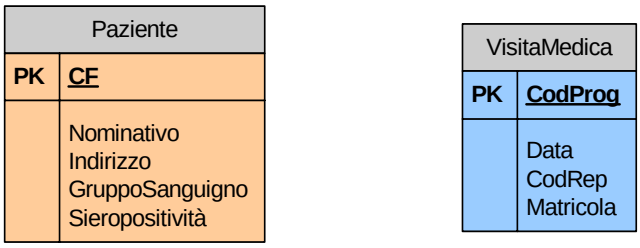
Esempio

Si consideri una tabella *Paziente* associata ad una tabella *VisiteMediche*. Nel modello logico è previsto che un singolo paziente possa effettuare molte (N) visite mediche, definite da istanze di *VisitaMedica* e costituite da tuple del tipo [CodProg(PK), Data, CodRep, Matricola] come risulta dall'esempio di Figura 5. Viceversa, una singola istanza

di *VisitaMedica* può contenere uno ed un solo collegamento a *Paziente*, essendo appunto che una visita medica viene fatta su un singolo paziente.

La PK di *Paziente*, rappresentata da CF, dovrà essere inserita come FK in *VisitaMedica*. In questo modo ogni istanza di *Paziente* "vedrà" molte (N) istanze di *VisitaMedica*, mentre viceversa un'istanza di *VisitaMedica* avrà come associata un'unica istanza di *Paziente*. In questo modo si andrà a realizzare un'associazione binaria 1:N correttamente gestibile dal DBMS.

... date due tabelle ...



... l'associazione 1:N viene specificata con ...

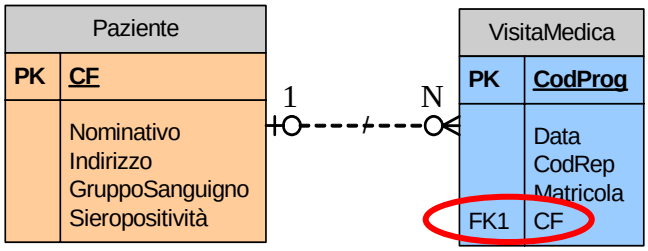


Figura 7: Esempio di associazione 1:N. Nella tabella "N" (VisitaMedica) viene inserita come FK1 la PK dalla tabella "1" (Paziente). Si noti la grafica che rappresenta l'associazione 1:N. Tale notazione viene denominata "a zampa di gallina".

CASO N:M (più complesso da gestire rispetto ai precedenti)

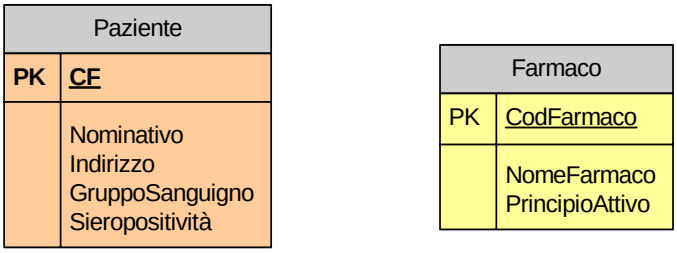
Esempio

Nel modello concettuale E/R di Figura 1 non erano definite associazioni N:M, quindi bisognerà rifarsi ad un altro esempio.

Si pensi ad una tabella *Pazienti* associata ad una tabella *Farmaco*, entrambe facenti parte di uno schema di base dati relativo ad un modello di gestione delle somministrazioni di farmaci ai pazienti di una struttura ospedaliera.

Anche se si tratta di un modello non analizzato in precedenza, è intuitivo pensare che ogni istanza di *Pazienti* "vede" molte istanze di *Farmaco*; basta ragionare pensando che nel periodo di ricovero un paziente può assumere a varie riprese più farmaci di diverso tipo.

Analogamente anche un'istanza di *Farmaco* "vede" molte istanze di *Paziente*, in quanto uno stesso farmaco può benissimo essere somministrato a molti pazienti differenti.
Viene a crearsi un'associazione binaria di tipo N:M.
Da un punto di vista logico questo è un problema, in quanto un siffatto tipo di associazione non consente una sua rappresentazione mediante il semplice inserimento di un attributo FK in una delle tabelle partecipanti.
La risoluzione di questa associazione N:M deve essere fatta introducendo nello schema logico una nuova tabella, denominata convenzionalmente *Tabella di associazione*, costituita almeno da due attributi FK1 e FK2 associati rispettivamente alle PK delle tabelle partecipanti. La PK della *Tabella di associazione* sarà semplicemente la combinazione (FK1, FK2).



... l'associazione N:M viene risolta con ...

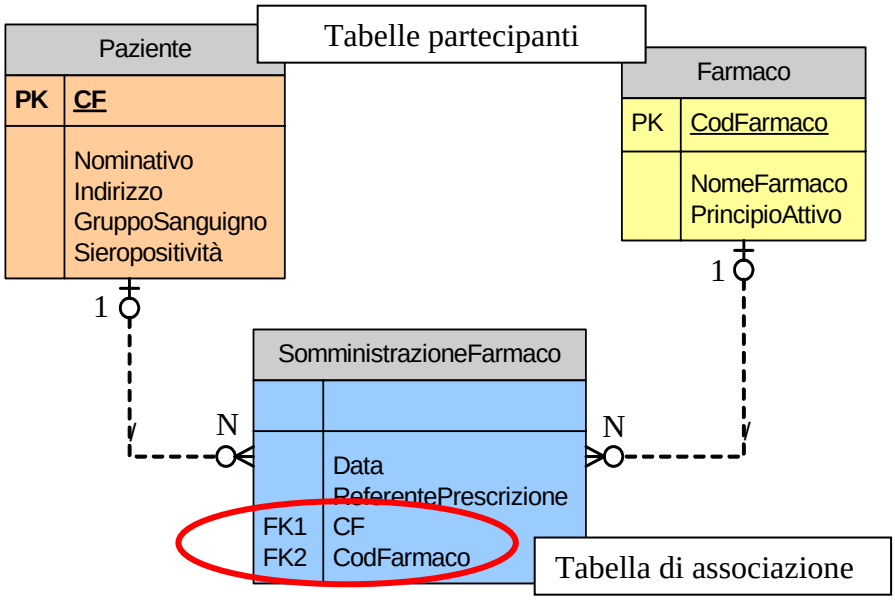


Figura 8: Esempio di associazione N:M. Deve essere inserita ex novo nel modello una tabella di associazione (SomministrazioneFarmaco) con almeno due attributi FK1 e FK2 collegati alle PK delle tabelle partecipanti. Nell'esempio, oltre ai due attributi necessari FK1 e FK2, per caratterizzare meglio le istanze di SomministrazioneFarmaco sono stati anche inseriti gli attributi Data e ReferentePrescrizione.

2.4.7 Un breve cenno all'algebra relazionale

Nell'esempio di Figura 5, esaminando la sola tabella VisitaMedica, non si riesce a dedurre immediatamente l'intera informazione contenuta in ogni sua istanza, in quanto le informazioni delle tabelle collegate sono di fatto codificate in ognuno dei valori assunti da FK. Tornando alla tabella VisitaMedica rappresentata in Figura 5, in riferimento ad esempio alla prima delle due istanze inserite, per interpretare compiutamente tutta l'informazione implicitamente contenuta nella tupla dovremmo eseguire una sorta di decodifica "manuale" dei termini mediante un confronto diretto con le rispettive tabelle collegate. La tupla selezionata "nasconde" infatti questa descrizione completa: *in data 20/06/2005, con codice progressivo di registrazione ORT20062005, è stata effettuata nell'U.O. Ortopedia (Ospedale Maggiore - Padiglione F) una visita medica alla sig.ra Bianchi Viviana da parte del dott. Verdi Gianni ortopedico (matr. 123D456).*

E' evidente che nella pratica, dovendo esplicitare le informazioni complete di tabelle contenenti migliaia o milioni di istanze, un tale approccio manuale risulterebbe inapplicabile. E' necessario quindi definire le operazioni matematiche che, gestite automaticamente dal DBMS, consentano di risolvere il problema di calcolo delle tuple risultanti dall'associazione di due o più tabelle tramite una serie di opportuni *operatori relazionali*. Pertanto il modello relazionale, oltre che definire la struttura di una base di dati, deve anche includere un insieme di operazioni che consentano la loro elaborazione matematica. Il citato insieme di operazioni è noto con il nome di *algebra relazionale*[10] e tra tutte le operazioni definite, in questa sede tratteremo solamente le due che appaiono necessarie per proseguire con i nostri ragionamenti: *il prodotto cartesiano* ed il *JOIN*[11].

2.4.8 Il prodotto cartesiano (tra tabelle)

L'operazione *prodotto cartesiano*, nota anche come *prodotto incrociato (cross product)* o *JOIN incrociato (cross JOIN)*, viene indicata con il simbolo × e fa parte delle cosiddette operazioni insiemistiche binarie.

Essa è usata per unire tuple prese da due tabelle in modo combinatorio. Date due tabelle $R(A_1, A_2, A_3, \ldots, A_n)$ e $S(B_1, B_2, B_3, \ldots, B_m)$, dove A_i e B_i sono i rispettivi attributi, il risultato di $R(A_1, A_2, A_3, \ldots, A_n) \times S(B_1, B_2, B_3, \ldots, B_m)$ è una nuova tabella Q con $n+m$ attributi $Q(A_1, A_2, A_3, \ldots, A_n, B_1, B_2, B_3, \ldots, B_m)$ presi in quest'ordine.

[10] Cfr. Nota 7.

[11] In realtà basterebbe definire il *JOIN*, ma bisogna necessariamente passare attraverso il *prodotto cartesiano* atteso che il primo deriva locicamente dal secondo.

Proviamo a capirci meglio con un semplice esempio.

Automobili (n=4)

A_1 A_2 A_3 A_4

Targa	Marca	Modello	FK CodColoreAuto
AB345FG	FIAT	Marea WE	2342
BC654GB	Ferrari	Quattroporte	3333
CD111FF	Ford	Focus	3333

ColoriCarrozzeria (m=2)

B_1 B_2

CodColore	Colore
2342	Giallo cromo
3333	Nero
5555	Bianco

Prodotto Cartesiano

AutomobiliColorate = Automobili × ColoriCarrozzeria (n+m=6)

Targa	Marca	Modello	CodColoreAuto		CodColore	Colore
AB345FG	FIAT	Marea WE	2342	⟷	2342	Giallo cromo
AB345FG	FIAT	Marea WE	2342		3333	Nero
AB345FG	FIAT	Marea WE	2342		5555	Bianco
BC654GB	Ferrari	Quattroporte	3333		2342	Giallo cromo
BC654GB	Ferrari	Quattroporte	3333	⟷	3333	Nero
BC654GB	Ferrari	Quattroporte	3333		5555	Bianco
CD111FF	Ford	Focus	3333		2342	Giallo cromo
CD111FF	Ford	Focus	3333	⟷	3333	Nero
CD111FF	Ford	Focus	3333		5555	Bianco

Figura 9: Risultato del prodotto cartesiano tra due tabelle Automobili e ColoriCarrozzeria. Si osservi che solamente le tre istanze selezionate rappresentano la reale corrispondenza tra le Automobili e i ColoriCarrozzeria collegate tramite la FK.

Osservando il risultato dell'esempio di Figura 9, si può intuire che *il solo prodotto cartesiano non presenta particolare utilità pratica*. Esaminate le istanze risultanti si nota infatti che in **AutomobiliColorate** *tutte le automobili hanno tutti i colori*, a prescindere dal valore assunto dalla FK in **Automobili**.

Nella pratica è molto più utile poter disporre di un operatore relazionale che, restando nell'esempio, ad un tipo di Automobili colleghi solo i Colori di carrozzeria associati e definiti dalla corrispondenza FK = PK. Per analogia si veda infatti l'altro esempio riportato

in Figura 5, dove *alcuni pazienti sono stati sottoposti a specifiche visite mediche da parte di specifici medici specialisti.*

Il risultato pratico che più interessa è quindi quello di ottenere un *sottoinsieme del prodotto cartesiano*, contenente tutte e sole le istanze derivanti da una *selezione del prodotto cartesiano basata sulla corrispondenza delle chiavi.*

Vista l'utilità pratica, è stato specificatamente definito un operatore che svolge esattamente questa funzione, denominato JOIN.

2.4.9 L'operazione JOIN

Senza dilungarci troppo, agganciandoci alle conclusioni del precedente capitolo 2.4.8 e riferendoci allo stesso esempio delle automobili, definiamo l'operazione di JOIN (indicata con $\bowtie$) nel modo seguente:

JOIN = Prodotto Cartesiano seguito da una Selezione basata sulla corrispondenza **CodColoreAuto = CodColore**

AutomobiliColorateCorrisp $\leftarrow$ **Automobili** $\bowtie_{\text{CodColoreAuto = CodColore}}$ **ColoriCarrozzeria**

Utilizzando le stesse tabelle e le stesse istanze di Figura 9, applicando un JOIN otteniamo il risultato desiderato rappresentato in Figura 10:

AutomobiliColorateCorrisp $\leftarrow$ **Automobili** $\bowtie$ **ColoriCarrozzeria (n+m=6)**

Targa	Marca	Modello	CodColoreAuto	CodColore	Colore
AB345FG	FIAT	Marea WE	2342	2342	Giallo cromo
BC654GB	Ferrari	Quattroporte	3333	3333	Nero
CD111FF	Ford	Focus	3333	3333	Nero

Figura 10: Risultato dell'operazione di JOIN tra due tabelle Automobili e ColoriCarrozzeria basata sulla corrispondenza CodColoreAuto = CodColore

2.4.10 I vincoli di integrità referenziale

Esaminando il modello logico di Figura 4, si può osservare che ogni istanza di una tabella associata dove compaiono uno o più attributi FK, stante proprio l'imprescindibile ruolo algebrico di questi ultimi (rappresentano infatti l'elemento che collega le tuple tra loro), deve sempre contenere un valore di FK $\neq 0$ e tale valore deve necessariamente trovare una corrispondenza nella/e relativa/e tabella/e correlata/e.

Per meglio chiarire il concetto, come sempre conviene rifarsi con un esempio prendendo la tabella VisitaMedica contenente le due istanze indicate in Figura 11.

Data	CodProg	CF	CodRep	Matricola
20/06/2005	ORT20062005	BNCVVN23G66F344Y	ORT4	123D456
12/03/2005	OTR12032005	GLLCRM01L45F443Y	OTR01	136H654

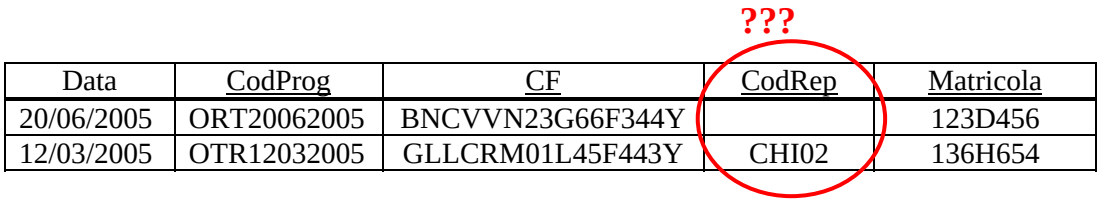
Figura 11: Tabella VisitaMedica con due istanze esemplificative

I valori ORT4 e OTR01 (FK della tabella VisitaMedica) collegano le due istanze ai due reparti dove le visite mediche sono state effettuate, rispettivamente U.O. Ortopedia e U.O. Otorinolaringoiatria, definiti a loro volta in Reparto con le PK ORT4 e OTR01, come evidenziato in Figura 12.

CodRep	Sede	Descrizione
OTR01	Ospedale Maggiore - Padiglione A	U.O. Otorinolaringoiatria
PDT04	Clinica universitaria	Cattedra di Pediatra
ORT4	Ospedale Maggiore - Padiglione F	U.O. Ortopedia

Figura 12: Tabella Reparto con tre istanze esemplificative

Partendo da questa premessa, è evidente che in VisitaMedica una serie di istanze come quelle indicate in Figura 13 non avrebbe senso, in quanto nella prima non c'è alcun collegamento al reparto dove la visita medica è stata effettuata e nella seconda il codice del reparto non trova corrispondenza nella tabella collegata Reparto (in sostanze il codice CHI02 nella PK di Reparto non c'è).



Data	CodProg	CF	CodRep	Matricola
20/06/2005	ORT20062005	BNCVVN23G66F344Y		123D456
12/03/2005	OTR12032005	GLLCRM01L45F443Y	CHI02	136H654

Figura 13: Tabella VisitaMedica con due istanze in cui il CodRep non permette l'individuazione dell'associazione con la tabella Reparti. Queste istanze violano i vincoli di integrità referenziale.

In tali casi si dice che istanze di questo tipo "violano i vincoli di integrità referenziale". In gergo, due istanze siffatte sono "orfane", cioè non "vedono" alcuna istanza nella tabella "padre" rappresentata in questo caso da Reparto. Esaminiamo ora altri esempi per illustrare meglio come si può arrivare a violare i vincoli di integrità referenziale.

Facendo riferimento alla tabella VisitaMedica di Figura 11, immaginiamo di entrare nella tabella Reparto ed eliminare la prima delle due istanze (quella del reparto U.O. Otorinolaringoiatria). Appare chiaro che, così facendo, ricreiamo sostanzialmente la

situazione già esaminata nell'esempio precedente, in quanto l'istanza afferente al reparto U.O. Otorinolaringoiatria diventa orfana.

Infine, sempre riferendoci alla tabella VisitaMedica di Figura 11, se in tabella Reparto modifichiamo il valore della PK (per esempio scrivendo OTR05-A al posto di OTR01), l'istanza collegata in VisitaMedica ridiventa orfana. Nel capitolo 3.2.5 verrà illustrata la procedura per impostare i vincoli di integrità referenziale in Microsoft Access®.

2.4.11 La normalizzazione delle tabelle

Arrivati quasi alla fine del capitolo sulla modellistica, appare necessario fare almeno un cenno su un aspetto molto importante della progettazione logica di un database: *la normalizzazione delle tabelle.* Finora abbiamo ipotizzato che gli attributi di una tabella vengano raggruppati per formare uno schema relazionale usando sostanzialmente *il buon senso e la capacità* del progettista. Invece, soprattutto per quanto concerne i database relazionali più complessi, la soggettività sulla realizzazione del modello logico ottimale lascia il posto ad un'analisi oggettiva, basata su rigorosi algoritmi matematici e definita *normalizzazione*, che mira sostanzialmente alla valutazione formale del perché un certo insieme di gruppi di attributi sia da considerarsi migliore di un altro.

Rimandando alla letteratura specialistica per tutti gli approfondimenti del caso[12], basti ricordare che le prime tre *Forme Normali*, denominate rispettivamente 1FN, 2FN e 3FN, sono sinteticamente definite secondo lo schema di Figura 14.

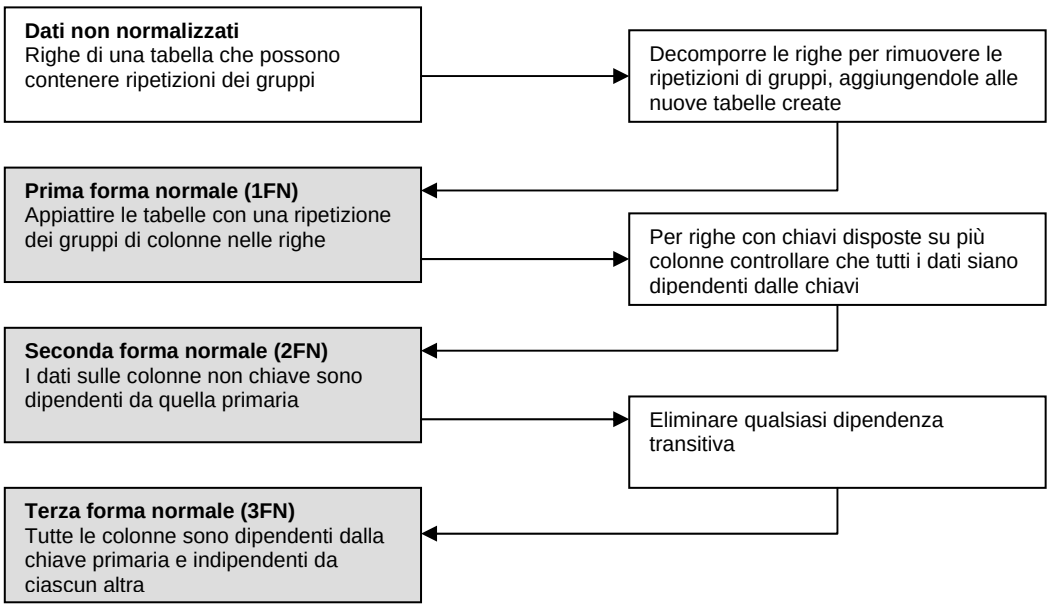


Figura 14: Definizione delle tre Forme Normali di Codd 1FN, 2FN e 3FN.

[12] Cfr. Nota 7

2.4.12 Esercizi

Esercizio 4
Si consideri la tabella Automobili dell'esempio di capitolo 2.4.8. In essa è stato scelto come PK l'attributo Targa. Immaginate di avere a disposizione, per ogni automobile, anche il NumeroTelaio.
Ritenete che anche questo attributo possa essere considerato PK ? Notate differenze pratiche rispetto la PK Targa ? Se sì, quali ? Se no, perché ?

Esercizio 5
In riferimento alla tabella di Figura 11, è possibile che più istanze abbiano lo stesso valore di CF ? Se sì, come interpretereste questa informazione ?

3 La scrittura del modello fisico ed il popolamento del database

Produrre il modello fisico del database equivale ad individuare specificatamente un DBMS commerciale che sappia tradurre in pratica lo schema di base dati fondato sulle operazioni di algebra relazionale descritte al capitolo 2.4.7.

Nel nostro caso, come già anticipato in premessa, il DBMS sarà Microsoft Access® e lo schema di base dati sarà quello descrittivo del caso studio esplicitato al capitolo 2.1 e rappresentato in Figura 1 e in Figura 4 rispettivamente nelle vesti concettuale E/R e logica relazionale di Codd.

3.1 Una breve panoramica descrittiva del software Microsoft Access®

Microsoft Access® è il DBMS commerciale di Microsoft e fa parte del pacchetto Office® assieme ai noti Microsoft Word® e Microsoft Excel®, con i quali garantisce un discreto livello di interscambiabilità dei dati.

Una delle comodità di Microsoft Access® rispetto altri DBMS di fascia analoga è quella di archiviare in un unico file con estensione "mdb" tutte le informazioni inerenti la banca dati, gli oggetti, le query (che definiremo in seguito) e gli eventuali moduli contenenti i codici di programmazione avanzata.

Il software dispone di una buona piattaforma per la realizzazione di comode interfacce utente compatibili con gli oggetti ActiveX, la nota tecnologia Microsoft che consente di personalizzare una vastissima gamma di comandi ed istruzioni mediante comodissimi oggetti programmabili dotati di specifiche proprietà e metodi dedicati.

Infine Microsoft Access® supporta il linguaggio di programmazione Visual Basic for Application (VBA), con il quale un programmatore anche non particolarmente esperto è in grado di realizzare interfacce di accesso ai dati dotate di automatismi estremamente efficaci.

3.1.1 L'interfaccia utente di Microsoft Access®

Figura 15: L'interfaccia utente iniziale di Microsoft Access®

In riferimento all'interfaccia utente iniziale di Microsoft Access® di Figura 15, sono individuabili i seguenti oggetti del database:

- **Tabelle**: rappresentano l'equivalente fisico delle tabelle (relazioni) logiche del modello relazionale e costituiscono il vero e proprio contenitore fisico dei dati.
- **Query**: sono lo strumento di interrogazione del database, mediante il quale è possibile estrarre in maniera selettiva le informazioni contenute nel database. Le query di Microsoft Access® supportano il linguaggio SQL.
- **Maschere**: sono lo strumento per la realizzazione delle interfacce utente necessarie per l'accesso guidato all'interno del database. Nelle maschere è possibile utilizzare la tecnologia ActiveX di Microsoft unitamente al linguaggio di programmazione VBA.
- **Report**: sono lo strumento per realizzare riepiloghi di stampa dei dati.
- **Pagine**: sono lo strumento per realizzare pagine di accesso al database in formato HTML esportabili direttamente nei browser web.
- **Macro**: sono lo strumento di base per la realizzazione di sequenze predefinite di comandi utili all'automatizzazione di alcune funzioni di accesso al database.

- **Moduli**: sono lo strumento che consente l'accesso all'interfaccia di VBA in cui è possibile scrivere codice di programmazione utile alla personalizzazione delle funzioni del database.

Considerati gli scopi della presente dispensa, ci limiteremo ad un'approfondita analisi delle caratteristiche di Microsoft Access® per quanto concerne la realizzazione delle Tabelle e, in misura inferiore, delle Query. La gestione delle Maschere verrà trattata unicamente per quelle che sono le informazioni essenziali e per i rimanenti oggetti, unitamente ad un approfondimento delle parti connesse con la programmazione VBA delle interfacce, si rimanda ad altra sede.

3.1.2 La Guida in linea di Microsoft Access®

Il taglio della presente dispensa, essendo incentrato soprattutto sugli aspetti della modellistica parzialmente svincolati dal DBMS scelto per lo sviluppo del modello fisico dei dati, non consentirà un dettagliato approfondimento delle questioni tecniche inerenti i comandi di Microsoft Access®.

A tal proposito merita quindi sottolineare che Microsoft Access® dispone di una completa e dettagliata Guida in linea, attivabile con il tasto F1 oppure con l'icona **?** posta nella barra principale degli strumenti.
Riguardo alla Guida in linea ci limiteremo a consigliarne vivamente l'utilizzo in tutti quei casi dove sussistano dubbi o conoscenze insufficienti dei comandi disponibili.

3.2 La realizzazione delle tabelle in Microsoft Access®

In coerenza con quanto abbiamo affermato a più riprese, questa fase pratica di realizzazione fisica delle tabelle presuppone un background sufficientemente approfondito inerente la cosiddetta modellistica, senza il quale il lettore potrebbe rischiare di imparare a memoria una serie di comandi senza però cogliere l'essenza della struttura logica di database relazionale, che di tutta la materia rappresenta una base imprescindibile.
A chi si appresta a continuare, consigliamo quindi un'attenta lettura e condivisione almeno di quanto riportato nel capitolo 2 della presente dispensa.

3.2.1 Partiamo "in manuale"

Microsoft Access® consente di realizzare le tabelle dello schema di base dati anche utilizzando delle procedure guidate semi-automatiche, che però riteniamo poco consigliabili soprattutto a chi per la prima volta ha un approccio con questo strumento.
Come indicato in Figura 16 partiremo per così dire "in manuale", con una procedura forse più lenta ma che sicuramente ci consentirà di seguire e controllare passo passo tutte le fasi del nostro percorso. Dopo aver selezionato la voce Crea una tabella in visualizzazione Struttura, cliccare sull'icona Struttura.

Figura 16: Creazione di una tabella in visualizzazione Struttura.

Con la conferma del comando Struttura compare l'interfaccia rappresentata in Figura 17, nella quale si devono impostare i campi della tabella, il relativo tipo di dati e la chiave primaria, cliccando sull'opportuna icona della barra degli strumenti (con il simbolo).
Nell'esempio riportato è stato inserito solamente il campo CF, stabilendo per esso il *Tipo dati* "Testo" con lunghezza massima di 16 caratteri, in coerenza con le caratteristiche della stringa alfanumerica del Codice Fiscale associato ad ogni persona fisica di nazionalità italiana. Si osservi che, a fianco della colonna *Tipo dati*, è presente una colonna *Descrizione* facoltativa, in cui è possibile (e consigliato) inserire alcune informazioni che servono a descrivere i dati relativi all'attributo in questione. Tali informazioni sono note con il termine di *metadati*. Nell'ovale è infine evidenziata la proprietà *Duplicati non ammessi* che viene attribuita automaticamente da Microsoft Access® al campo PK.

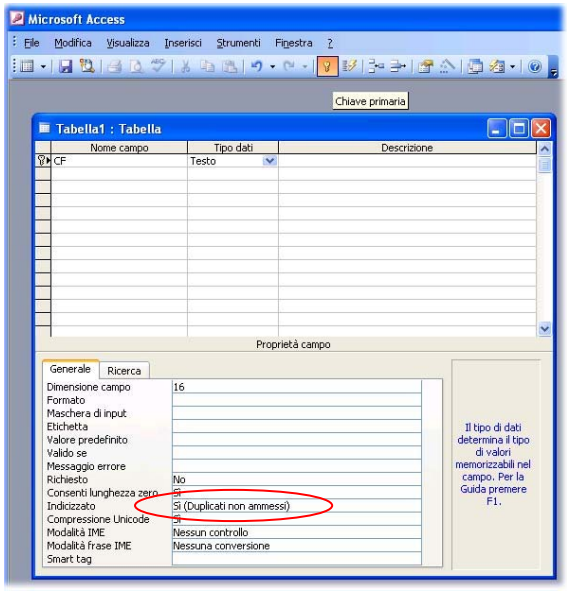


Figura 17: Interfaccia di impostazione degli attributi (Nome campo), della Chiave primaria e di tutte le caratteristiche del Tipo dati da associare ad ognuno degli attributi.

3.2.2 Un approfondimento sul *Tipo dati*

Il modello fisico di database prevede la precisa definizione del formato dei dati afferenti ad ogni attributo delle tabelle. Microsoft Access® supporta una serie di *Tipo dati* così costituita:

Testo

Si utilizza quando nel campo dovranno essere immessi valori alfanumerici (numeri o lettere) generalmente aventi lunghezza massima di 255 caratteri (spazi compresi).

Memo

Si utilizza in quei campi di tipo descrittivo, dove c'è la necessità di inserire stringhe alfanumeriche anche molto lunghe (intere frasi), tipiche per esempio dei campi "Note". Il formato supporta stringhe alfanumeriche di lunghezza massima di 65.535 caratteri.

Numerico

Si utilizza nei campi dove dovranno essere inserite solo stringhe numeriche, utilizzate per eseguire calcoli di varia natura. Sono disponibili molte tipologie di formato numerico.

Data/ora

Si utilizza in quei campi dove si devono inserire dati di tipo data/ora, come per esempio date di nascita, scadenze, appuntamenti, ecc.

Valuta

Si utilizza in quei campi dove i numeri assumono il significato di valuta.

Contatore

E' un formato numerico (intero lungo) molto importante, utilizzato nella maggior parte dei casi per i campi PK. La peculiarità più importante di questo formato numerico è che Access®, automaticamente, gestisce un contatore incrementale intero che assicura l'irripetibilità del dato, caratteristica peculiare delle PK.

Si/No

E' il tipico formato booleano, utilizzabile in tutte le variabili dicotomiche. In tabella viene registrato il valore 0 per No ed il valore -1 per Sì.

Oggetto OLE

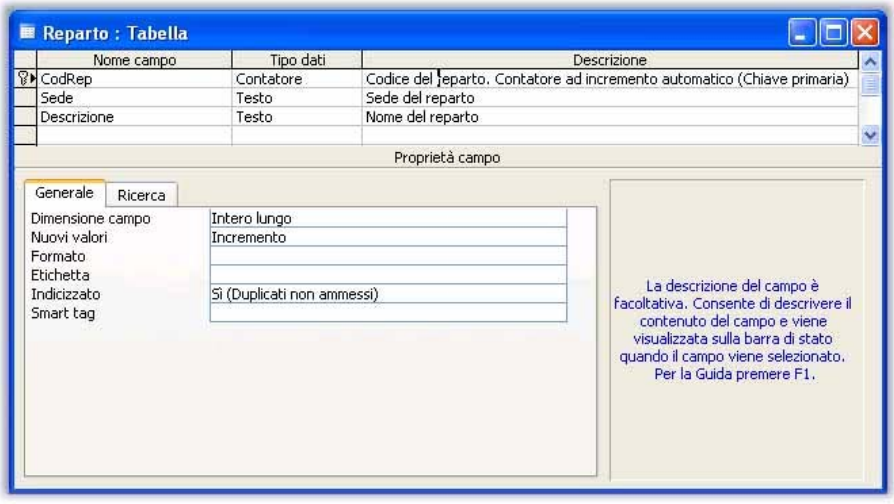
Si utilizza in quei campi dove il contenuto non è di tipo alfanumerico. Un oggetto OLE è per esempio un file di immagine (una fotografia, un filmato, ecc.).

Collegamento ipertestuale

E' il formato che si utilizza quando la stringa del campo rappresenta un collegamento ipertestuale verso un altro file.

3.2.3 Il risultato finale di realizzazione delle tabelle

Sulla base del modello logico relazionale rappresentato in Figura 4 e tenuto conto dei concetti esposti nei capitoli 3.2.1 e 3.2.2, le quattro tabelle costituenti il modello fisico del caso studio, complete di attributi e della loro descrizione (metadati), sono riportate in Figura 18.



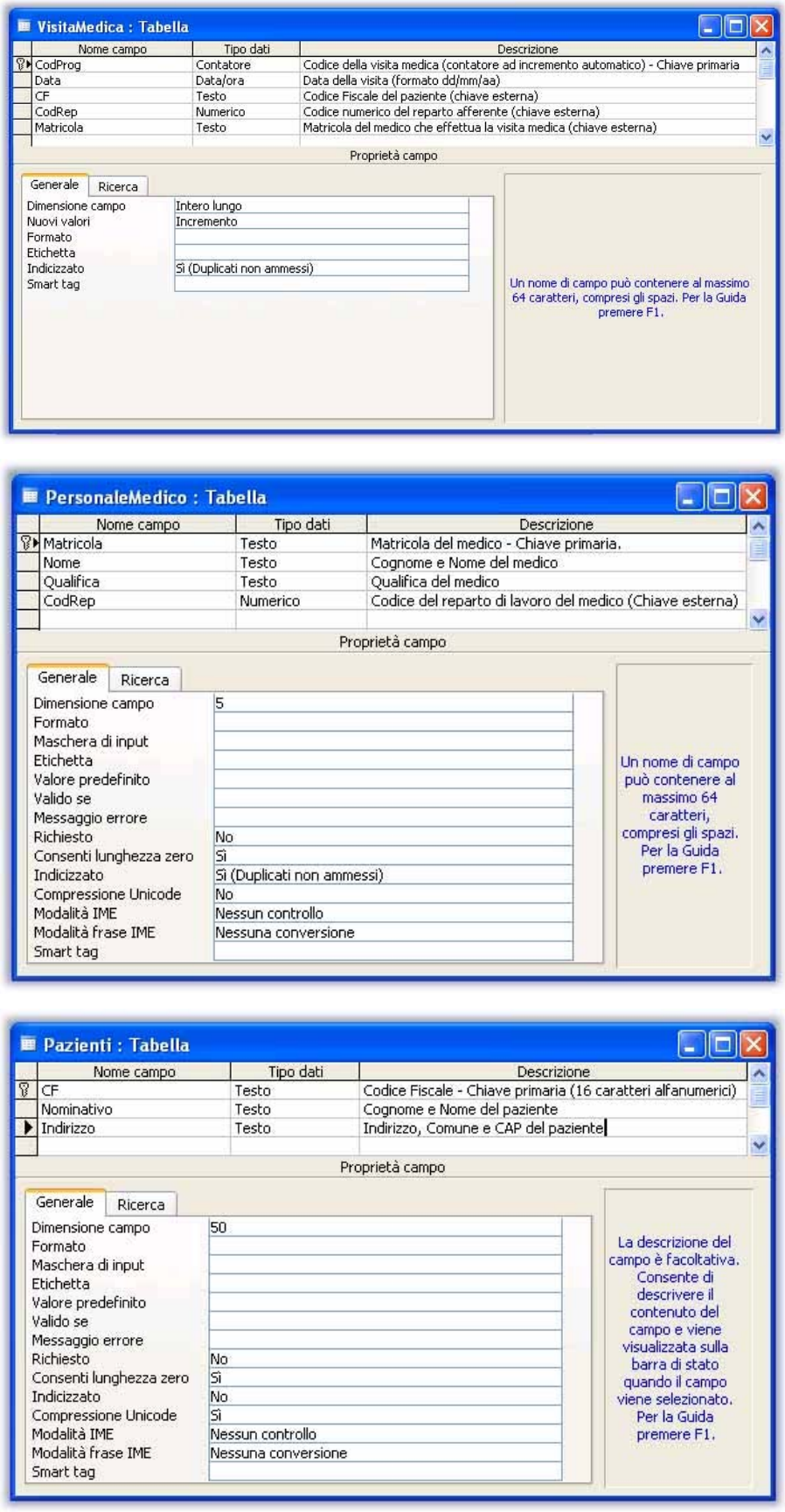


Figura 18: Le quattro tabelle del modello fisico dei dati complete di metadati.

3.2.4 La definizione delle associazioni binarie (JOIN) con Microsoft Access®.

Come già sottolineato ai capitoli 2.4.5 e 2.4.6, il modello logico è costituito dalle tabelle e dalle associazioni. Quindi nel modello fisico, dopo la realizzazione delle tabelle con inserimento di tutte le FK necessarie, determinate in funzione della cardinalità delle associazioni binarie, è necessario procedere alla definizione "fisica" delle suddette associazioni. Ricordiamo che nel modello logico le associazioni corrispondono a precise operazioni matematiche binarie denominate JOIN (Cfr. capitolo 2.4.9).

Con Microsoft Access® questa procedura viene eseguita mediante il comando Relazioni come visualizzato in Figura 19. Riconosciamo una certa imprecisione nella scelta del nome del comando, che più propriamente avrebbe dovuto chiamarsi Associazioni. Ciò deriva sempre dalla impropria traduzione del termine inglese Relationship, che nel contesto in questione assume il significato di Associazione e non di Relazione. Nella versione inglese di Microsoft Access® il comando citato si chiama infatti più correttamente Relationship.

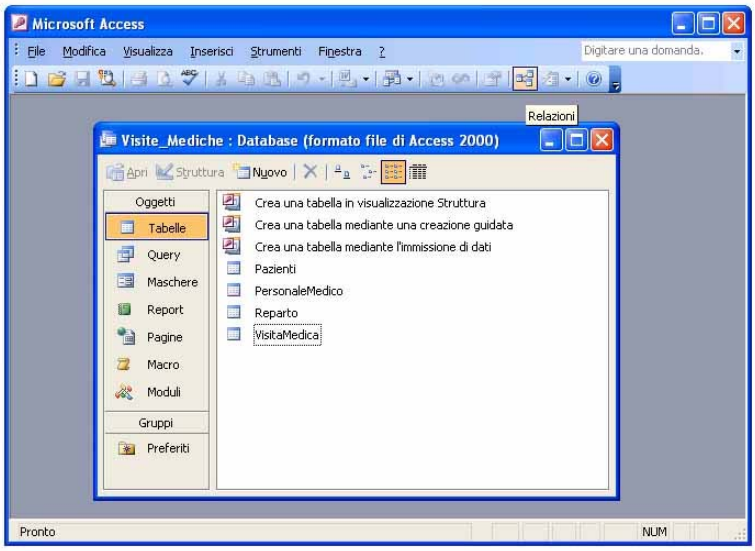


Figura 19: Comando per l'attivazione dell'interfaccia *Relazioni* necessaria per la definizione delle associazioni (JOIN) tra le tabelle del database.

E' necessario selezionare una alla volta le tabelle che devono essere associate tra loro cliccando di volta in volta il tasto Aggiungi. In Figura 20 è riportato il risultato finale della procedura di aggiunta delle tabelle da porre in associazione tra loro.

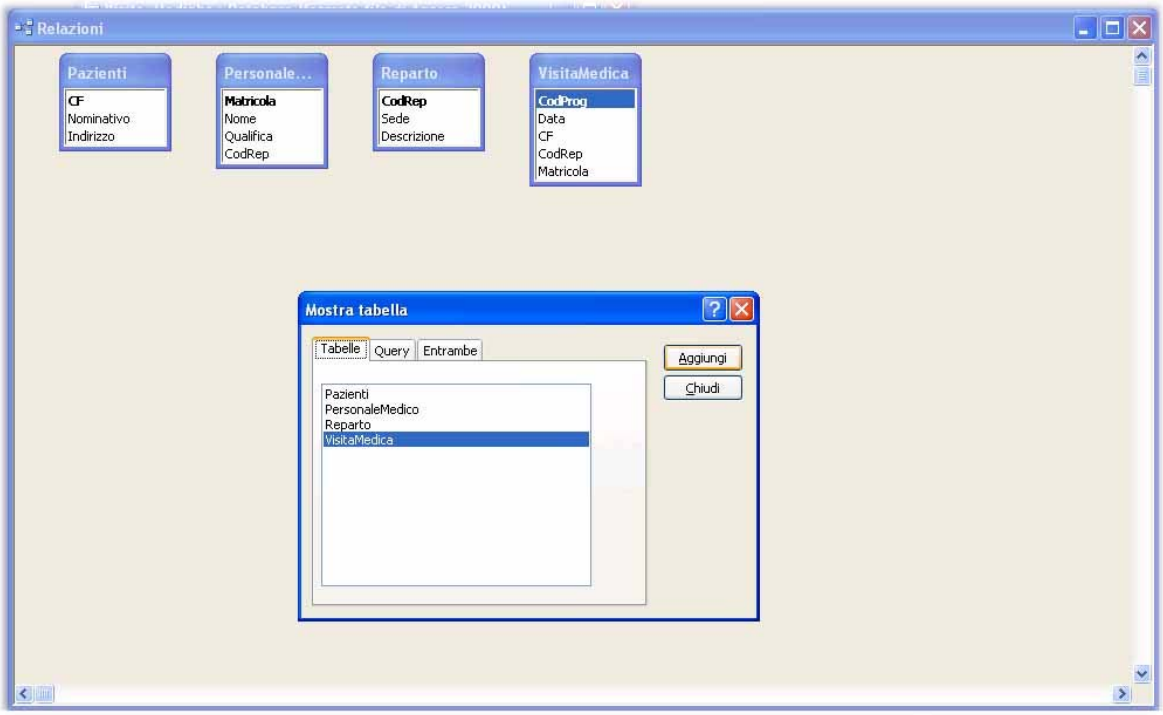


Figura 20: Interfaccia per la definizione delle associazioni binarie tra le tabelle del modello fisico.

Selezionando e trascinando ogni chiave primaria PK delle tabelle (1) sulla corrispondente chiave esterna FK delle tabelle (N), si generano automaticamente i JOIN, visualizzati mediante una linea nera come mostrato in Figura 21. Per maggiore chiarezza, in Figura 22 riportiamo lo stesso schema realizzato con una simbologia grafica che rende evidente la cardinalità tra le varie associazioni.

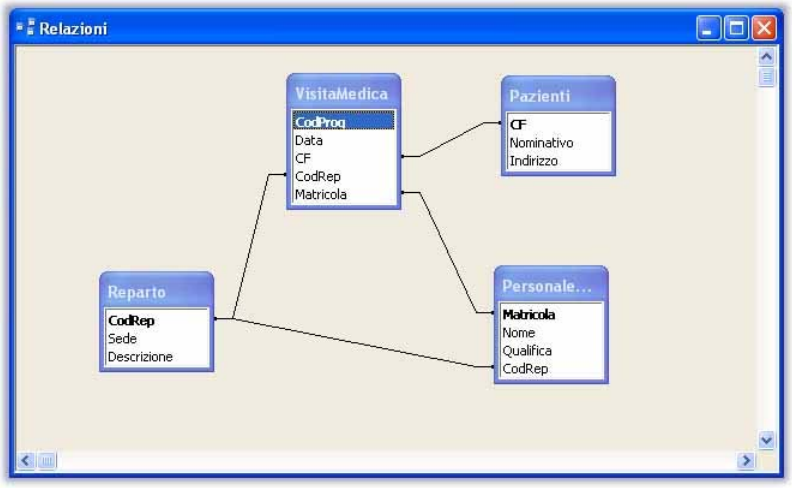


Figura 21: L'interfaccia di Microsoft Access® delle associazioni con il risultato finale della definizione dei JOIN. Si osservi come ogni chiave primaria PK sia stata associata alla relativa chiave esterna FK della tabella VisitaMedica.

In questa fase pratica di realizzazione dei JOIN ci permettiamo di segnalare un piccolo limite di Microsoft Access®, derivante dal fatto che, contrariamente a quanto accade in

molti altri DBMS in cui il trascinamento di un campo PK dentro una tabella associata comporta l'inserimento automatico della corrispondente FK, in Microsoft Access® ciò non accade. Infatti la FK deve essere già inserita preventivamente all'atto della realizzazione di ogni tabella "figlio" (lato N).

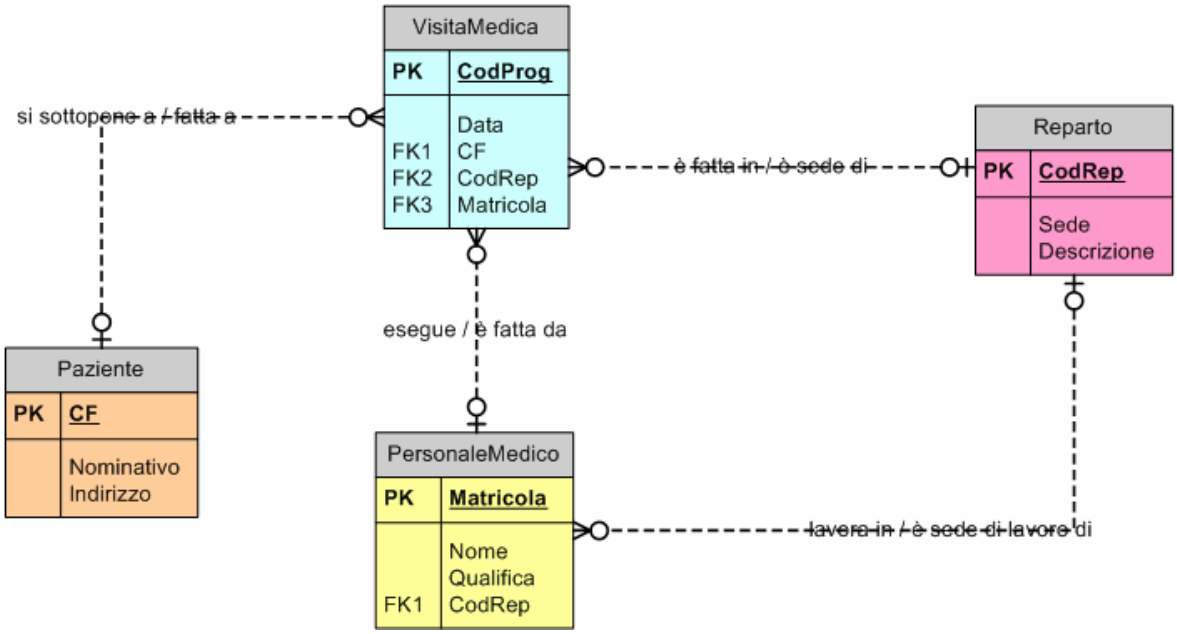


Figura 22: Il modello logico relazionale con indicazione della cardinalità tra le associazioni binarie rappresentata con la notazione "a zampa di gallina". Per chiarezza sono riportati i verbi (Cfr. Figura 4).

3.2.5 Impostazione dei vincoli di integrità referenziale in Microsoft Access®

In riferimento al capitolo 2.4.10, nel quale era stato definito il concetto di "vincolo di integrità referenziale", per impostare queste proprietà in Microsoft Access® è sufficiente entrare nell'interfaccia Associazioni di Figura 21 e fare doppio click sopra la linea nera che rappresenta graficamente l'associazione in cui si desidera impostare un vincolo di integrità referenziale. Così facendo viene visualizzata l'interfaccia di Figura 23, in cui si possono appunto selezionare le opzioni desiderate.

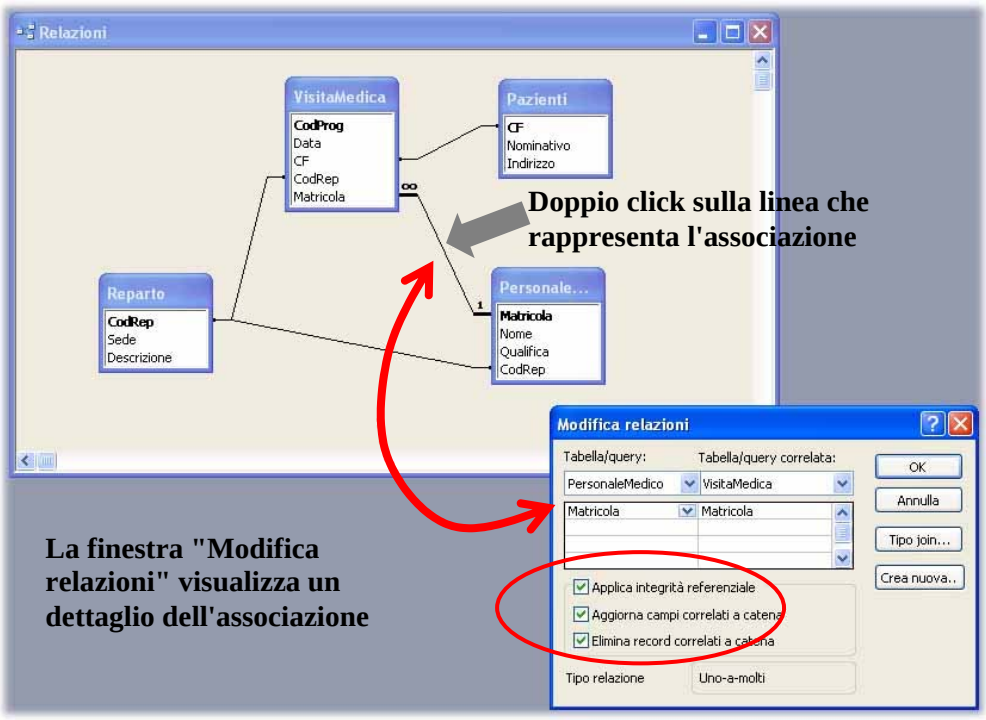


Figura 23: Impostazione dei vincoli di integrità referenziale in Microsoft Access®. E' possibile scegliere i tipi di vincoli di integrità da applicare all'associazione scelta (check box cerchiate). Si osservi che, una volta impostato almeno uno dei vincoli di integrità referenziale, sul simbolo grafico della relativa associazione compaiono le notazioni 1 e ∞ (rispettivamente nel "lato 1" e nel "lato N").

3.3 Il popolamento del database (direttamente dalle tabelle)

Completate le operazioni di scrittura fisica del modello di database secondo le procedure descritte nei capitoli precedenti, è finalmente possibile passare alla fase vera a propria di inserimento dati, chiamata anche convenzionalmente *popolamento del database*.

Con Microsoft Access® il popolamento può essere fatto direttamente dall'interfaccia tabelle anche se, come accennato al capitolo 3.1.1, lo strumento più adatto offerto dal software è l'oggetto *maschera* del cui funzionamento faremo qualche cenno al capitolo 5.

3.3.1 L'importante è partire dalla tabella giusta

Si è parlato al capitolo 2.4.10 dei vincoli di integrità referenziale, in particolare del fatto che un database relazionale non deve ammettere istanze orfane.

Il popolamento quindi non può prescindere da ciò, pertanto esso dovrà tenere conto di tutte le istanze logicamente collegate tra loro.

Per spiegarci meglio, soprattutto per quanto concerne il *primo popolamento* (quello per intenderci che parte dal database vuoto), non sarà ammissibile iniziare l'inserimento di istanze in VisitaMedica se prima non saranno state inserite istanze nelle tabelle "padre"

rappresentate da Reparto, PersonaleMedico e Paziente. Per comprendere la logica di questa sequenza, basterà ricordare che il popolamento delle tabelle "figlio" non può prescindere dall'aver precedentemente popolato tutte le tabelle "padre".
Quindi inizieremo a popolare le tabelle Paziente e Reparto, per poi passare alla tabella PersonaleMedico ed infine alla tabella VisitaMedica. E' curioso notare che la tabella forse più rappresentativa, dove sono registrate proprio le istanze che sostanziano il caso studio "registrazione degli accessi dei pazienti", è proprio quella che maggiormente dipende dalle istanze delle altre. Questo fatto non deve sorprendere più di tanto, in quanto è intuitivo pensare anche concettualmente che le istanze di una tabella "figlio" (com'è VisitaMedica) derivino appunto da una o più istanze delle tabelle "padre".

3.3.2 L'inserimento dati mediante l'interfaccia Tabella

In riferimento alla Figura 19, in base alla sequenza stabilita al precedente capitolo 3.3.1, fare doppio click sull'icona della tabella Paziente. Dall'interfaccia di Figura 24 sarà possibile procedere all'inserimento delle istanze che, per semplicità, saranno quelle già utilizzate negli esempi di capitolo 2.4.4.

Figura 24: Inserimento delle istanze in tabella Paziente utilizzando direttamente l'interfaccia Tabella.

Per quanto concerne la tabella Reparto, la procedura di inserimento dati sarà identica a quella appena descritta. Viceversa il popolamento delle tabelle "figlio" PersonaleMedico (contiene la FK CodRep) e VisitaMedica (contiene le FK CodRep, CF, Matricola), dovrà essere fatto avendo cura di riportare nelle FK i rispettivi valori (già) inseriti nelle corrispondenti tabelle "padre".

3.3.3 Gli effetti pratici dei vincoli di integrità referenziale di aggiornamento ed eliminazione a catena delle istanze correlate

L'applicazione di una qualsiasi delle regole di integrità referenziale, come mostrato in Figura 23, induce specifiche azioni da parte di Microsoft Access®, che in caso di eliminazione di un'istanza collegata risponde con il messaggio di Figura 25.

Figura 25: Messaggio di Microsoft Access® derivante dall'impostazione della regola di integrità referenziale "eliminazione record correlati a catena".

3.3.4 La violazione di un vincolo di integrità referenziale e la risposta di Microsoft Access®

In riferimento alla Figura 23, nella quale sono stati impostati 3 vincoli di integrità referenziale, proviamo a vedere cosa accade se inseriamo un'istanza "orfana di PersonaleMedico" nella tabella VisitaMedica.

Consideriamo dapprima le istanze (già) inserite nella tabella PersonaleMedico, rappresentate in Figura 26. Osserviamo che i valori delle PK Matricola nelle tre registrazioni sono rispettivamente 123D456, 234G567 e 136H654.

Figura 26: Inserimento delle istanze in tabella PersonaleMedico utilizzando direttamente l'interfaccia Tabella.

Proviamo ora ad inserire un'istanza in tabella VisitaMedica in cui, forzando volutamente il vincolo di integrità referenziale applicato, mettiamo nel campo FK Matricola il valore (inesistente!) 222F123.

Il DBMS, "accortosi" della violazione del vincolo, risponderà con il messaggio di errore rappresentato in Figura 27, impedendoci di inserire l'istanza orfana.

Figura 27: Messaggio di errore di Microsoft Access® derivante dalla forzatura di un vincolo di integrità referenziale.

3.3.5 Il risultato finale

Per chiudere riportiamo in Figura 28 le tabelle del modello fisico finale contenenti le istanze esemplificative inserite, che torneranno utili come riferimento per gli esercizi svolti nel successivo capitolo 6.

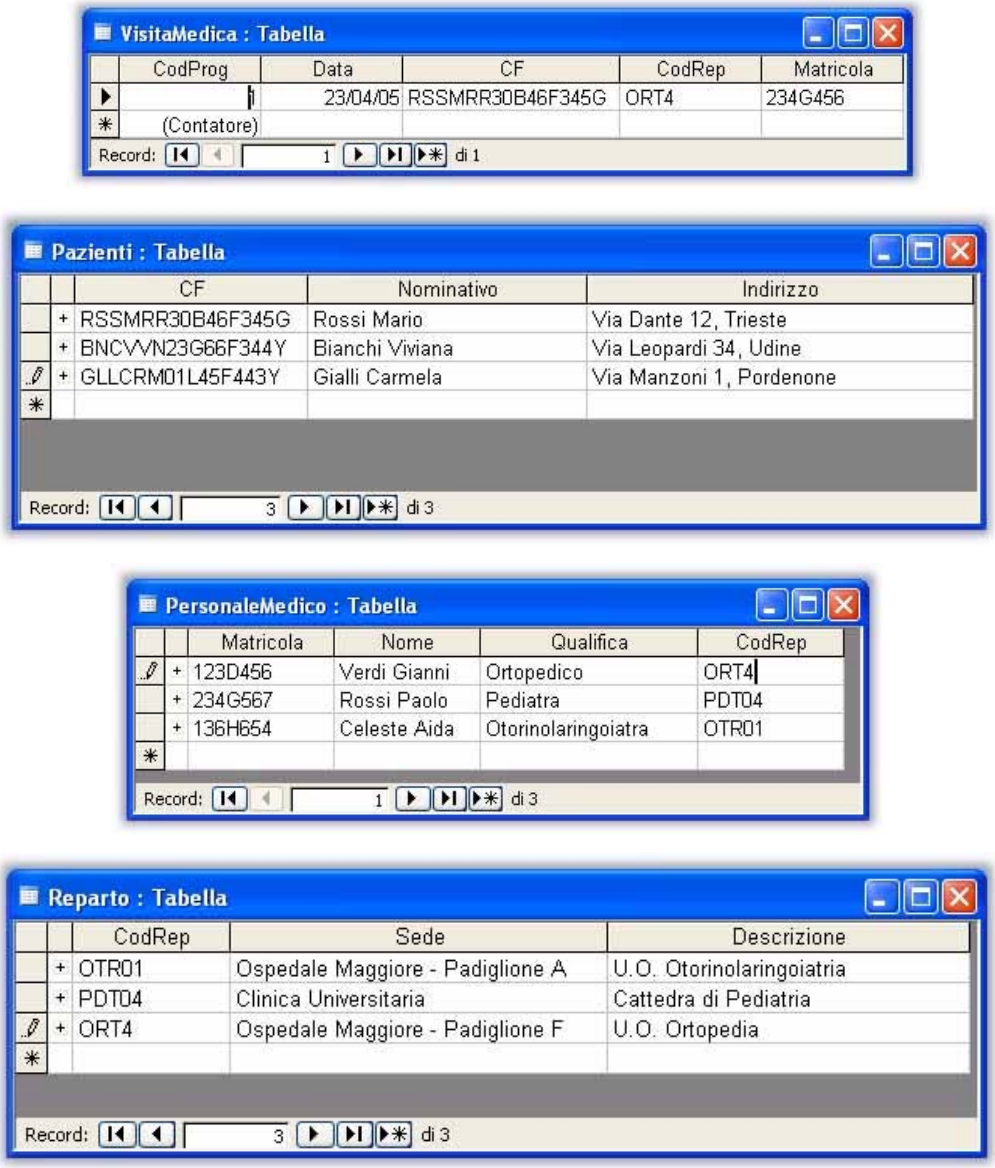


Figura 28: Le quattro tabelle popolate del modello fisico finale realizzate con Microsoft Access®

4 L'interrogazione del database: le query

4.1 Il linguaggio SQL (cenni)

Nel capitolo 2.4.7, esaminando le operazioni di algebra relazionale *prodotto cartesiano* e *JOIN*, erano stati illustrati alcuni metodi matematici che consentivano di interagire con le tabelle ottenendo interessanti risultati di notevole utilità pratica.

Uno dei limiti delle operazioni matematiche dell'algebra relazionale consiste nel fatto che la logica e la sintassi delle regole dipendono dal DBMS scelto per lo sviluppo del modello, con ovvie implicazioni (negative) in caso di necessità di migrazione della struttura di base dati da un DBMS ad un altro.

Al fine di porre rimedio a questa lacuna, nel 1986 l'American National Standards Institute (ANSI)[13] e l'ISO hanno raccolto ed uniformato le diverse versioni di un linguaggio universale sviluppato tra gli anni '70 ed '80 dal Centro Ricerche IBM denominato Structured Query Language e noto in tutto il mondo con l'acronimo SQL, diventato oggi il linguaggio standard per i DBMS commerciali e supportato anche da Microsoft Access®.

In Microsoft Access® le stringhe SQL vengono realizzate mediante l'oggetto *query*, che offre una comoda interfaccia user friendly con la quale, paradossalmente, è possibile costruire complesse stringhe SQL ... senza conoscere praticamente nulla di SQL, come vedremo più avanti.

SQL è un linguaggio completo, comprendente cioè le istruzioni per la definizione dei dati, quelle per la loro interrogazione e quelle per il loro aggiornamento.

[13] American National Standards Institute: *"The Database Language SQL"*, Document ANSIX3.135, 1986.

In questa sede ci occuperemo solamente della parte di SQL relativa alle istruzioni di interrogazione della base dati.

4.1.1 La sintassi generale di una stringa di interrogazione SQL

La sintassi delle interrogazioni fondamentali di SQL è basata sulle tre clausole SELECT, FROM e WHERE raggruppate nel cosiddetto blocco SELECT-FROM-WHERE (Figura 29).

Il blocco ha la seguente forma:

SELECT <elenco attributi>
FROM <elenco tabelle>
WHERE <condizione>

dove:

- <elenco attributi> è un elenco dei nomi degli attributi i cui valori devono essere recuperati dall'interrogazione.
- <elenco tabelle> è un elenco dei nomi delle tabelle necessarie per eseguire l'interrogazione.
- <condizione> è un'espressione condizionale (opzionale) che identifica le tuple che devono essere recuperate dall'interrogazione (in sostanza è la stringa che definisce il filtro).

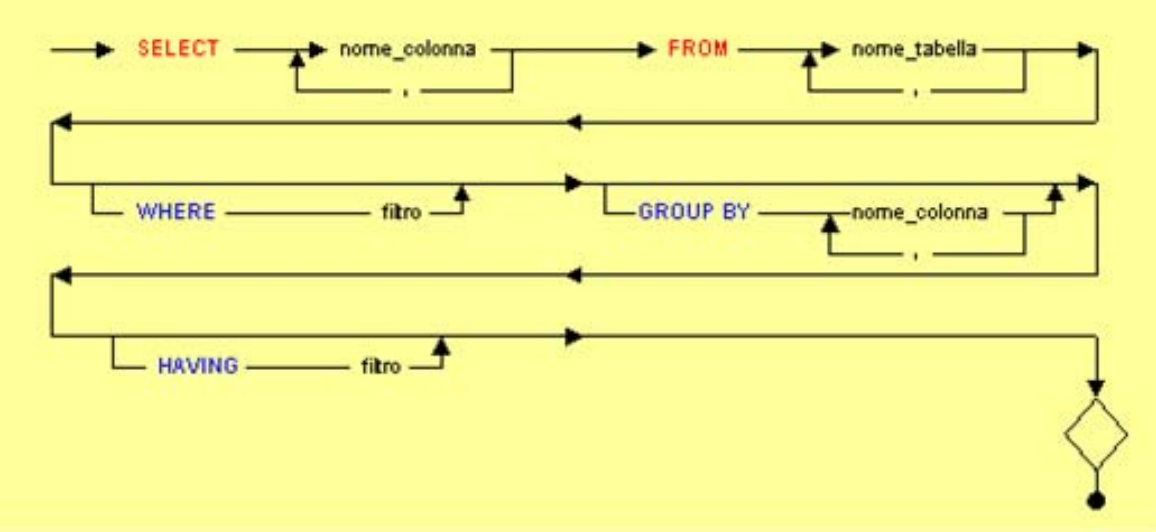


Figura 29: Diagramma del blocco SELECT-FOM-WHERE di SQL®. In rosso sono indicate le clausole indispensabili, mentre in blu quello opzionali.

4.1.2 Esempi di stringhe SQL

Per chiarezza faremo riferimento al modello fisico già popolato e presentato al capitolo 3.3.

Interrogazione 1

Estrarre l'indirizzo del paziente con CF = "GLLCRM01L45F443Y"

La stringa SQL assumerà la seguente forma:

SELECT Indirizzo
FROM Paziente
WHERE CF="GLLCRM01L45F443Y"

Interrogazione 2

Estrarre i pazienti che hanno effettuato visite mediche nel reparto di ortopedia.

La stringa SQL assumerà la seguente forma:

SELECT CF, Nominativo, Indirizzo, Descrizione, CodProg, Data
FROM Paziente, Reparto, VisitaMedica
WHERE CodRep="ORT4"

4.2 La realizzazione delle query in Microsoft Access®

Quasi sempre una delle componenti principali che concorre a decretare il successo commerciale di un software, soprattutto se questo è indirizzato al grande pubblico, è la sua semplicità d'uso. Basti pensare al salto di qualità (anche commerciale!) avvenuto con il passaggio dall'interfaccia utente del sistema operativo MS DOS a quella di Microsoft Windows®.

In questa logica, riteniamo che l'interfaccia offerta da Microsoft Access® per la realizzazione delle query, anche a conferma di quanto affermato al capitolo 1.3 "*Prerequisiti (sottotitolo: Access è facile)*", sia sicuramente un degno esempio di semplicità.

Le query che traducono nel modello fisico le stringhe SQL di interrogazione del database sono rappresentate in Microsoft Access® dalle *query di selezione* e dalle *query a campi incrociati*, delle quali appunto descriveremo le procedure di realizzazione nei capitoli che seguono.

4.2.1 Partiamo "in manuale" con una semplice query di selezione

Come già sperimentato con le tabelle al capitolo 3.2.1, anche per la realizzazione delle query conviene subito rifarsi con un esempio pratico.

Immaginiamo quindi di voler realizzare la query corrispondente alla stringa SQL del caso Interrogazione 1 trattato al capitolo 4.1.1. E' una semplice stringa di estrazione che, in Microsoft Access®, prende il nome di query di selezione.

Come indicato in Figura 30 partiremo per così dire "in manuale", con una procedura forse più lenta ma che sicuramente ci consentirà di seguire passo passo tutte le fasi del nostro percorso.

Dopo aver selezionato all'interno dell'oggetto Query la voce Crea una query in visualizzazione Struttura, cliccare sull'icona Struttura.

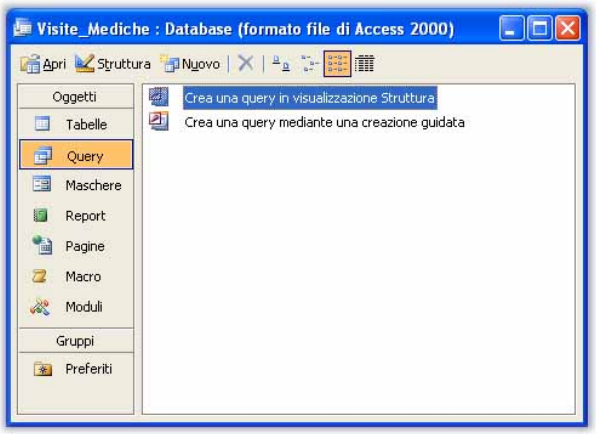


Figura 30: Creazione di una query in visualizzazione Struttura.

Con la conferma del comando Struttura compare l'interfaccia rappresentata in Figura 31, nella quale si devono inserire le tabelle in cui la query dovrà recuperare i dati[14], che nell'esempio considerato sarà l'unica tabella Paziente.

[14] Volutamente, qui come nel seguito, non faremo riferimento alle clausole della stringa SQL, volendo dimostrare che in Microsoft Access® non è necessario conoscere la sintassi SQL per realizzare una query.

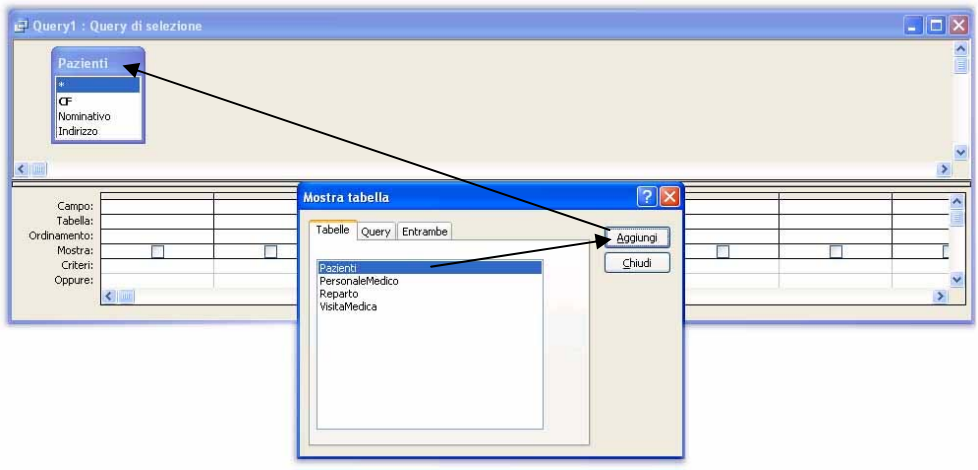


Figura 31: Aggiunta di una (o più) tabella (e) nell'interfaccia di realizzazione di una query

Una volta inserita la tabella, da essa sarà necessario procedere alla selezione degli attributi (campi) che la query dovrà visualizzare.

In riferimento alla Figura 32, anche in questo caso la procedura risulta estremamente semplice ed infatti basterà fare doppio click sui campi desiderati affinché questi siano automaticamente inseriti nella struttura della query come intestazione di colonna.

L'ultimo passo consiste nell'impostazione del filtro, ovvero del criterio sul quale si basa l'estrazione dei dati desiderati. A tale scopo sarà sufficiente scrivere nella riga Criteri in corrispondenza del campo CF la stringa "GLLCRM01L45F443Y".

La query è finita, non resta che eseguirla cliccando semplicemente il pulsante ! posto sulla barra dei comandi ed automaticamente Microsoft Access® restituisce il risultato dell'estrazione, come mostrato in Figura 33.

Una volta realizzata, la query può essere salvata per un suo riutilizzo futuro. A questo proposito, per quanto riguarda la scelta del nome nella fase di salvataggio, si rimanda al capitolo 4.2.3.

4.2.2 Ma dov'è la stringa SQL ?

Se l'interrogazione del database in Microsoft Access® presuppone un comando basato su una stringa SQL avente una precisa sintassi, deve essere altrettanto vero che la procedura di realizzazione "visuale" della query di selezione descritta al precedente capitolo 4.2.1, deve aver prodotto, da qualche parte, questa stringa.

Infatti, cliccando semplicemente sul comando SQL indicato in Figura 34, compare l'interfaccia con il testo della stringa cercata, visualizzata in Figura 35.

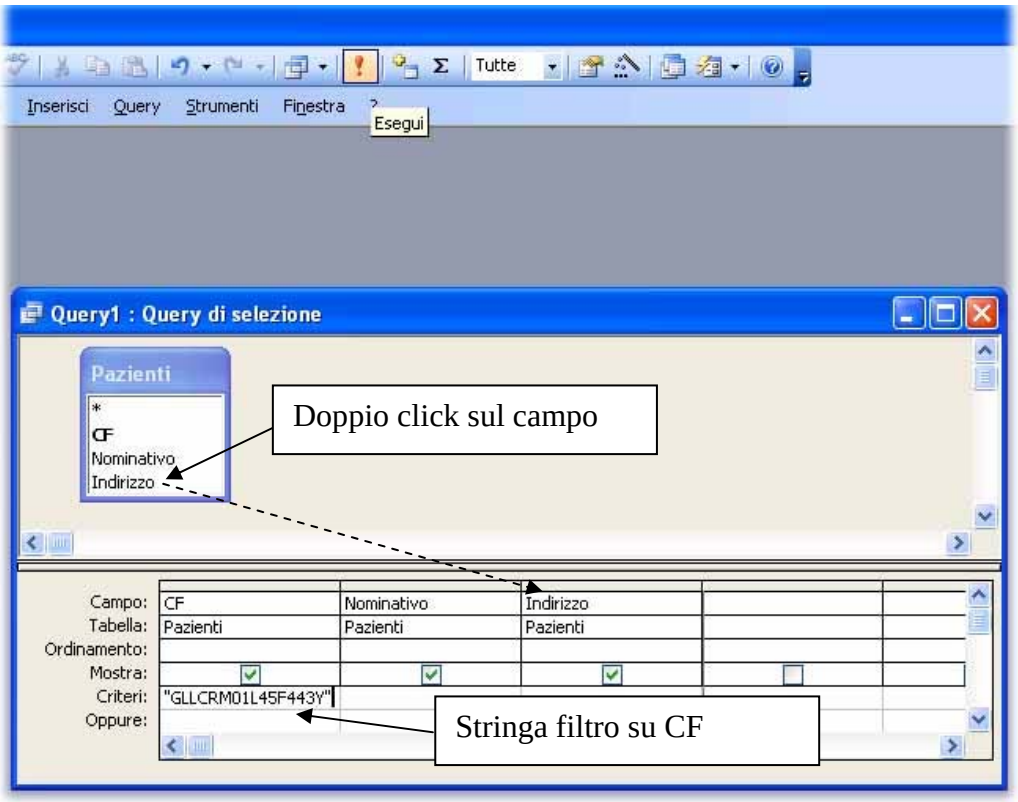


Figura 32: Impostazione dei parametri di una query di selezione basata su una tabella

Figura 33: Visualizzazione del risultato della query di selezione di Figura 32

Per molti potrà sembrare banale, però merita ricordare che in tutti i linguaggi di programmazione, SQL incluso, scrivere correttamente la sintassi di un comando significa rispettare tutta una serie di regole, comprese le virgole, i punti, le virgolette e gli altri eventuali simbolismi richiesti. L'omesso rispetto anche di una sola delle regole di sintassi comporta quasi sempre il mancato funzionamento del comando.

Quindi ci sembra doveroso affermare che la disponibilità offerta da Microsoft Access® di una comoda e semplice interfaccia per realizzare una query senza bisogno di conoscere

SQL, tra l'altro anche abbastanza intuitiva sotto il profilo concettuale, rappresenta un indubbio vantaggio per il programmatore che in questo modo è libero di concentrarsi molto di più sui contenuti piuttosto che sulla forma dei suoi ragionamenti.

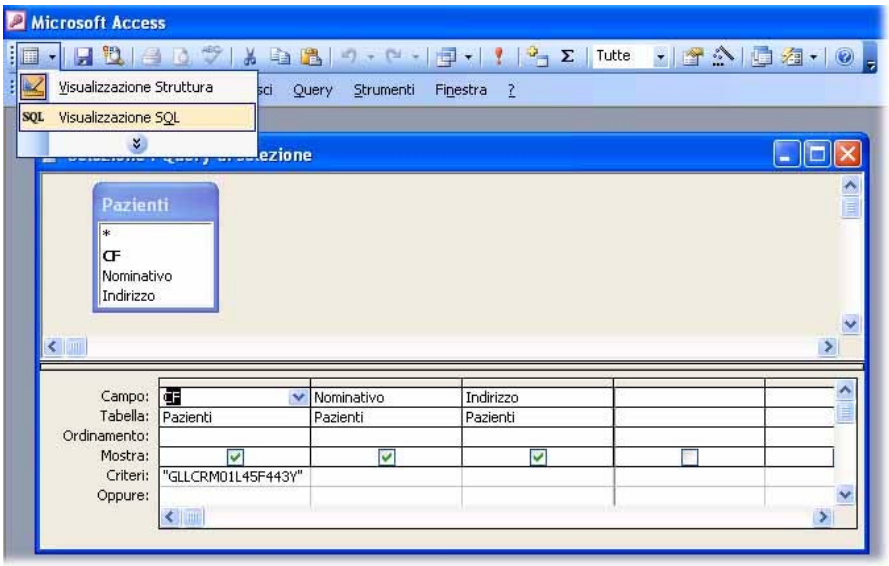


Figura 34: Il comando SQL che consente la visualizzazione della stringa SQL relativa alla query visualizzata in struttura

Figura 35: Visualizzazione della stringa SQL della query di Figura 34. Si noti, trattandosi di una query di selezione che interroga il database, il blocco costituito dalle tre clausole SELECT, FROM, WHERE.

4.2.3 Convenzioni stilistiche sulla scelta dei nomi delle query (e delle tabelle)

Osservando la Figura 33 ci si accorge immediatamente che il risultato di una query è una tabella di valori, del tutto simile nella forma e nei contenuti alle tabelle fisiche incontrate nel capitolo 3.2.

A dire il vero, concettualmente parlando, differenze tra le tabelle (che solo adesso conviene chiamarle nuovamente relazioni) e le query prodotte da SQL ce ne sono parecchie. Una su tutte: una relazione è un insieme, quindi in quanto tale non ammette duplicati. Viceversa il

prodotto di un'interrogazione SQL non lo è, quindi nell'elenco delle tuple risultanti possono benissimo comparire duplicati.
Ma, a parte quanto appena detto, chiunque può notare molte similitudini tra una tabella e un risultato di query. A conferma di ciò, come vedremo nel capitolo 5, l'oggetto origine di una maschera potrà essere scelto indifferentemente in una tabella oppure in una query. Così facendo, è bene che fin da subito si evitino confusioni nella scelta dei nomi da attribuire sia alle tabelle sia alle query. Il metodo convenzionale tradizionalmente impiegato, che ci permettiamo di consigliare vivamente, è quello di anteporre le stringhe TBL_ al nome delle tabelle e QRY_ al nome delle query. Nell'esempio trattato nella presente dispensa converrà quindi utilizzare i seguenti nomi: TBL_Paziente, TBL_VisitaMedica, TBL_PersonaleMedico_, TBL Reparto per le tabelle e, ad esempio, QRY_EstraiGialli per la query di selezione trattata al capitolo 4.2.1.

4.2.4 Una query di selezione più complessa

Restando nell'ambito delle query di selezione, proviamo ora a risolvere il caso relativamente più complesso inerente Interrogazione 2 esaminato al capitolo 4.1.1.
Seguendo la medesima procedura già descritta in precedenza, ricordando ora che le tabelle da inserire sono tre e non una, il risultato finale è quello indicato in Figura 36.
Appare necessario un doveroso commento su questo esempio.
Servendoci delle definizioni già date al capitolo 2.4.6, si noti che le tabelle Pazienti e Reparto sono legate da un'associazione N:M. Per tradurre questa associazione è stata conseguentemente inserita la tabella VisitaMedica che spezza tale associazione generandone altre due, entrambe in rapporto di cardinalità 1:N, mediante due operatori algebrici di JOIN.
Si osservi altresì che l'obiettivo dell'interrogazione rende di fatto inutili i campi della tabella VisitaMedica (di essa non è stato infatti inserito nessuno nella query), però l'inserimento della tabella stessa si è reso comunque necessario affinché l'operazione algebrica potesse essere risolta. Il mancato inserimento della tabella VisitaMedica avrebbe comportato la restituzione di un semplice prodotto cartesiano Pazienti×Reparto, non rispondente ai requisiti dell'interrogazione.
In Figura 37 ed in Figura 38 sono visualizzati rispettivamente la stringa SQL della query di Figura 36 (non proprio semplice, concordate ?) ed suo risultato di esecuzione.

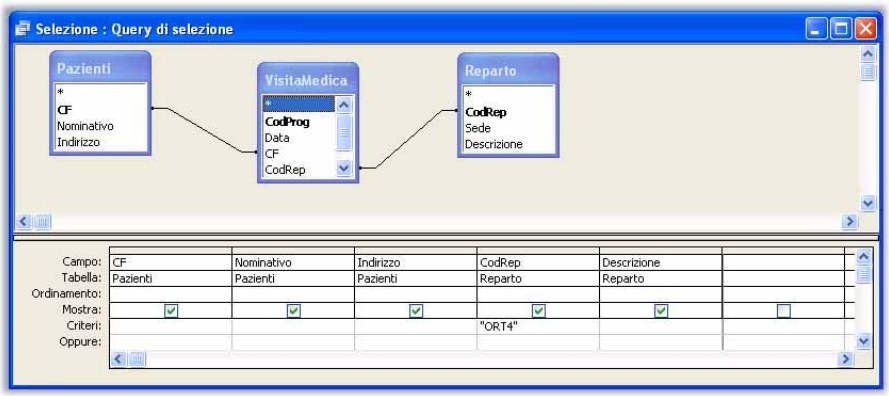


Figura 36: Impostazione di una query di selezione basata su più tabelle. Si noti che la procedura di inserimento delle tabelle importa automaticamente anche le eventuali associazioni

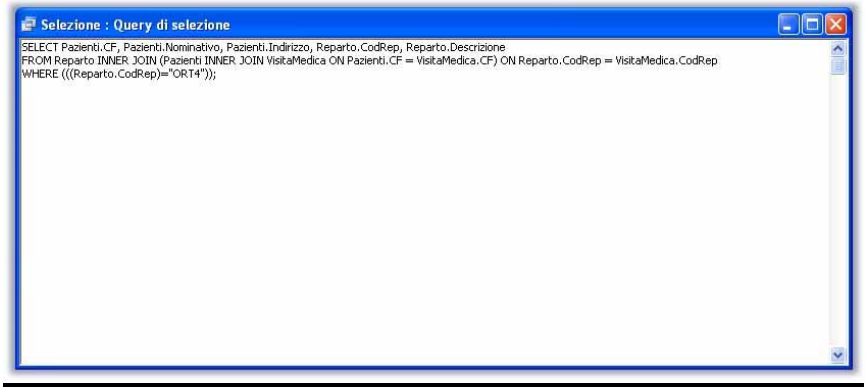


Figura 37: Visualizzazione della stringa SQL della query di Figura 36

Figura 38: Visualizzazione del risultato della query di selezione di Figura 36

4.2.5 Le query a campi incrociati

Con lo strumento *query a campi incrociati* Microsoft Access® offre la possibilità di realizzare interrogazioni al database con restituzione dei risultati in forma aggregata. Proviamo a spiegarci meglio con un esempio.

Interrogazione 3

Estrarre il numero di visite mediche effettuate da tutto il personale medico suddivise per reparto.

Non essendo necessaria, omettiamo di riportare la stringa SQL risultante dall'interrogazione, in quanto sarà Microsoft Access® a produrla dopo l'impostazione visuale della relativa query.

Il vero problema sul quale ci soffermiamo è invece quello di focalizzare meglio il quesito e capire *come* si desidera vengano restituiti i dati cercati. Crediamo sia condivisibile una restituzione espressa nella forma rappresentata in Figura 39, chiamata appunto *a campi incrociati*. E' infatti intuitivo pensare che l'interrogazione formulata presupponga che il sistema passi in rassegna tutto il personale medico e tutti i reparti, incrociando (da cui il nome) i rispettivi valori per restituire, laddove esistente, il risultato del conteggio degli incroci, corrispondente appunto al numero delle visite mediche effettuate da ogni medico in ogni reparto.

Campo *Intestazione di Riga*

Campo *Intestazione di Colonna*

Campo *Valore*

	Reparto 1	**Reparto 2**	**Reparto 3**	**...**	**Reparto j**	**...**	**Reparto n**
Medico 1	ContaVisite 1,1	ContaVisite 1,2	ContaVisite 1,3	...	ContaVisite 1,,j	...	ContaVisite 1,n
Medico 2	ContaVisite 2,1	ContaVisite 2,2	ContaVisite 2,3	...	ContaVisite 2,j	...	ContaVisite 2,n
Medico 3	ContaVisite 3,1	ContaVisite 3,2	ContaVisite 3,3	...	ContaVisite 3,j	...	ContaVisite 3,n
...	...	...	...	...	...	...	...
Medico i	ContaVisite i,1	ContaVisite i,2	ContaVisite i,3	...	**ContaVisite i,j**	...	ContaVisite i,n
...	...	...	...	...	...	...	...
Medico m	ContaVisite m,1	ContaVisite m,2	ContaVisite m,3	...	ContaVisite m,j	...	ContaVisite m,n

Figura 39: Schema concettuale di una query a campi incrociati con calcolo del conteggio del valore VisiteMediche sulla base dell'incrocio tra medici e reparti.

La procedura di costruzione della query a campi incrociati chiamata a risolvere Interrogazione 3, senza utilizzare l'autocomposizione per lo stesso motivo già ribadito più volte, prevede una prima fase di inserimento delle tabelle necessarie all'interno di una nuova struttura di query di selezione, come riportato in Figura 41. Successivamente, mediante il comando Query a campi incrociati posto nel menù principale, è necessario impostare la funzione *conteggio* nella riga Formula in corrispondenza del campo CodProg. Il risultato finale è rappresentato in Figura 41.

A titolo puramente informativo, si sappia che anche Microsoft Excel® dispone di un comando analogo alla query a campi incrociati, denominato Tabella Pivot e presente all'interno del menu Dati (Figura 40).

Figura 40: Maschera per la creazione guidata Tabella pivot e grafico pivot di Microsoft Excel®

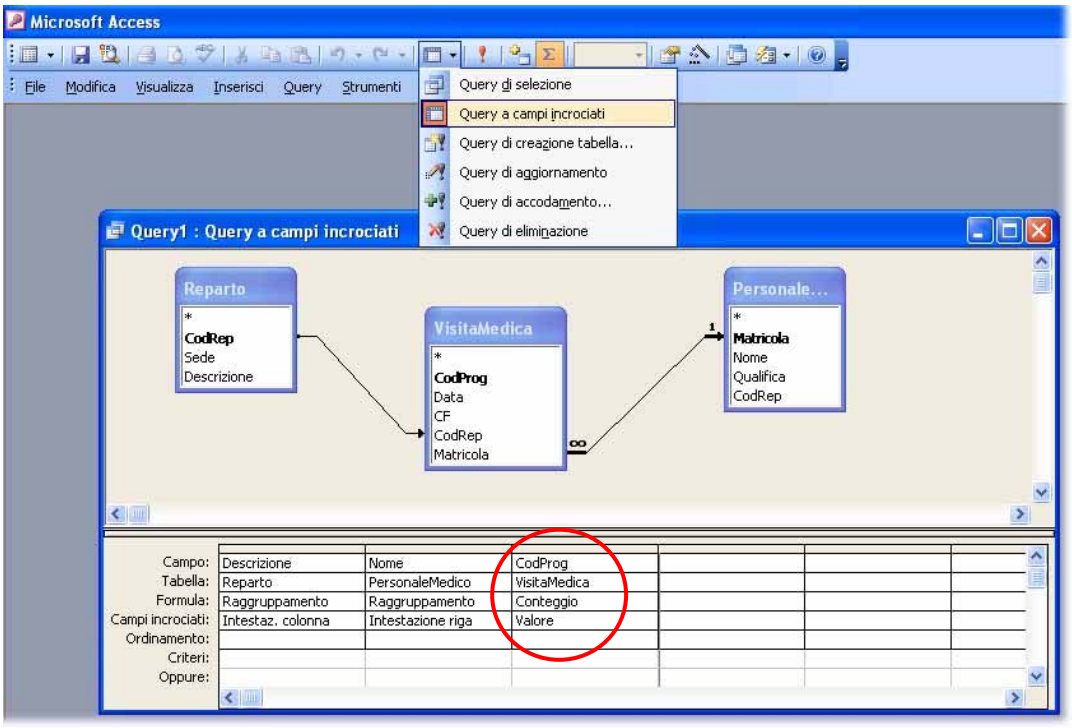


Figura 41: Impostazione dei parametri di una query a campi incrociati. Si osservino le righe *Formula* e *Campi incrociati*, comparse dopo la selezione del tipo di query effettuata con il comando del menù principale.

4.3 Procedure di scambio dati tra Microsoft Access® e Microsoft Excel®

Uno degli elementi che caratterizza il pacchetto Microsoft Office®, del quale si è fatto cenno al capitolo 3.1, è la possibilità di interscambio dei dati prodotti dai diversi applicativi costituenti il pacchetto stesso.

Nel presente capitolo tratteremo sinteticamente le tecniche di importazione ed esportazione dei dati da Microsoft Excel® a Microsoft Access® e viceversa. Non bisogna dimenticare, infatti, che i due applicativi citati rappresentano strumenti evoluti per eseguire determinate operazioni; il primo specializzato nei calcoli mentre il secondo nella gestione dei dati. Il buon utilizzatore deve quindi capire fin dove è conveniente utilizzare uno e da che punto in poi diventa invece comodo impiegare l'altro, onde massimizzare i benefici derivanti dalle rispettive specificità.

Il problema è quindi duplice: avendo a disposizione un insieme di dati in Microsoft Access® (tabelle o query), questi come possono essere esportati in Microsoft Excel® per poter effettuare lì ulteriori elaborazioni statistiche più avanzate ? Oppure: disponendo di una tabella in Microsoft Excel®, come si effettua una sua importazione in Microsoft Access® per renderla parte di uno schema di base dati ?

4.3.1 Da Microsoft Access® a Microsoft Excel®

Sfruttando quanto già realizzato negli esempi trattati, si desidera esportare in Microsoft Excel® la query di selezione rappresentata in Figura 36.

Dopo aver aperto una cartella di lavoro in Microsoft Excel® in cui importare i dati, selezionare in Microsoft Access® l'icona della query desiderata (non aprire la query!), cliccare Copia, spostarsi nella cartella di Microsoft Excel® all'interno del foglio di lavoro scelto e, da lì, cliccare Incolla (in Excel!).

4.3.2 Da Microsoft Excel® a Microsoft Access®

Per evitare problemi, conviene definire il nome dell'intervallo dati di Microsoft Excel® da esportare in Microsoft Access® seguendo le procedure specifiche riportate nella guida in linea (in Microsoft Excel® premere F1 e digitare la parola chiave *definire nome*, scegliendo successivamente la voce *modificare o eliminare un nome definito*).

In Microsoft Access® selezionare il comando File → Carica dati esterni → Importa, avendo cura di impostare nell'interfaccia che segue il tipo file *Microsoft Excel* come mostrato in Figura 42, confermando con il pulsante Importa.

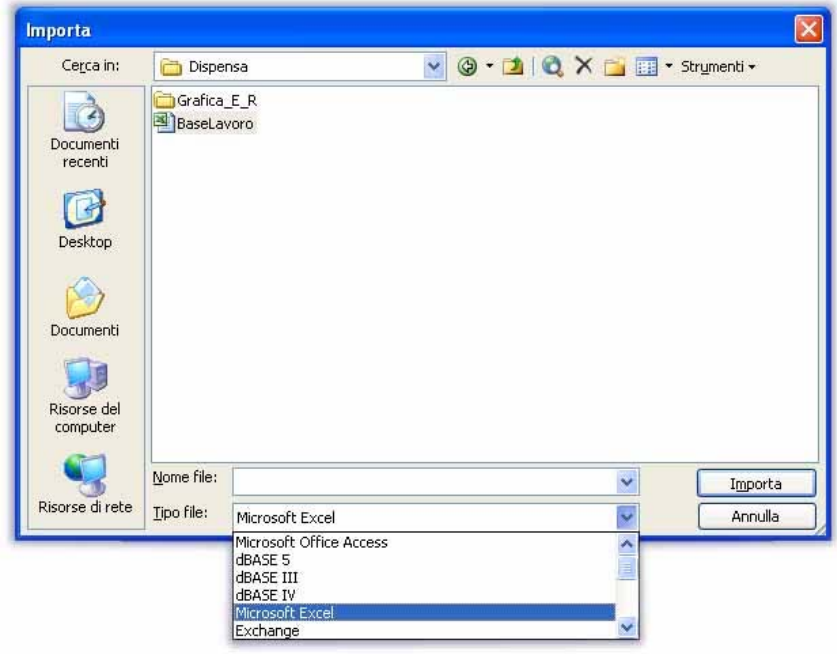


Figura 42: Importazione dei dati da Microsoft Excel®

La successiva importazione viene sviluppata mediante una procedura guidata che consente di scegliere tra le varie opzioni disponibili al fine di ottimizzare la qualità del dato finale. Si ricorda che una tabella importata non necessariamente possiede da subito tutte le caratteristiche che consentono di inserirla immediatamente nello schema di base dati, ma spesso (quasi sempre) sarà necessario procedere ad un suo editing soprattutto per quanto concerne la definizione delle chiavi, le quali, come noto, sono una prerogativa del modello relazionale supportato da Microsoft Access® ma non da Microsoft Excel®.

4.3.3 Esercizi

Esercizio 6
In riferimento al caso **Interrogazione 2** del capitolo 4.2.1, realizzare la query di selezione che visualizzi, oltre ai campi già indicati, anche il nominativo del medico che effettua le visite.

Esercizio 7 (impegnativo)
In riferimento al caso **Interrogazione 2** del capitolo 4.2.1, realizzare una query che visualizzi tutti i medici della tabella PersonaleMedico e, per quelli che hanno svolto visite mediche, i dati delle stesse rappresentati dai campi Data, Paziente, Reparto.

Suggerimento: costruire la query di selezione come indicato negli esempi già trattati e provare a fare doppio click sui simboli di JOIN ...

Esercizio 8
In riferimento alla query a campi incrociati relativa al caso **Interrogazione 3**, si spieghi il motivo per cui è stato scelto il campo CodProg di VisitaMedica per effettuare il conteggio del numero di visite mediche. Scegliendo il campo Data avremmo ottenuto lo stesso risultato ?
Suggerimento: il campo CodProg è PK di VisitaMedica, quindi ...

5 Le maschere (cenni)

Microsoft Access® dispone di una buona piattaforma per la realizzazione di efficaci interfacce utente compatibili con gli oggetti ActiveX, la nota tecnologia Microsoft che consente di personalizzare una vastissima gamma di comandi ed istruzioni mediante comodissimi oggetti programmabili in VBA e dotati di specifiche proprietà e metodi dedicati.

La trattazione esaustiva di questa materia richiederebbe uno spazio a sé che, visti gli scopi che si siamo prefissati nella presente dispensa, limiteremo al minimo rimandando alla vasta manualistica di approfondimento disponibile sul mercato o in internet[15]. Ce ne scusiamo in anticipo con il lettore, che forse rimarrà un po' deluso essendo forse questa la parte più "divertente" di Microsoft Access®.

5.1 Creazione di una maschera in visualizzazione struttura

A titolo di esempio descriveremo la procedura per realizzare una maschera per interagire con la tabella Pazienti.

Dal menu principale, selezionato l'oggetto Maschera, scegliere il comando Nuovo ed impostare in Visualizzazione Struttura la tabella Pazienti come origine dati (Figura 43), confermando con OK.

Figura 43: Creazione di una nuova maschera in visualizzazione struttura avente come origine dati la tabella Pazienti

[15] Prague, C.N., Irwin, M.R.,: *Microsoft Access 2000*, (IDG Books), Tecniche nuove, 1999

Dall'interfaccia di Figura 44, procedere al trascinamento dei campi all'interno del corpo della struttura della nuova maschera, che verranno automaticamente convertiti in oggetti ActiveX a seconda della loro impostazione nella struttura della tabella.

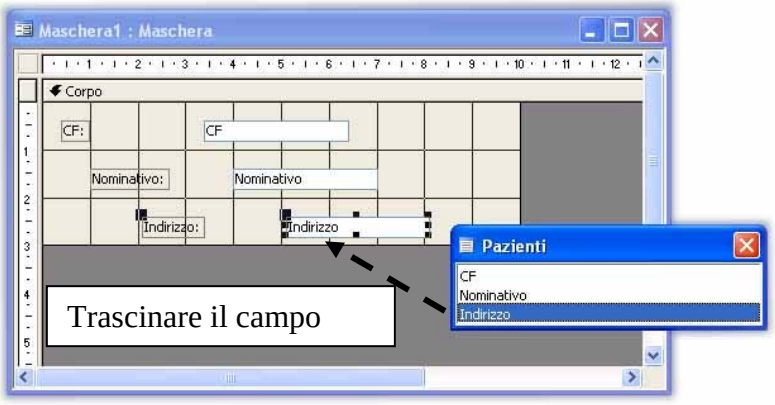


Figura 44: Trascinamento dei campi della tabella all'interno del corpo di struttura della nuova maschera

Terminata questa operazione, per rendere operativa la maschera basterà cliccare sul pulsante Visualizza della barra dei comandi principale. La maschera consentirà di visualizzare una per una le istanze della tabella di origine; accedendo alle caselle di testo sarà possibile aggiungere e/o modificare i dati.

5.1.1 La casella degli strumenti e le proprietà degli oggetti

L'inserimento degli oggetti ActiveX può avvenire anche mediante la casella degli strumenti rappresentata in Figura 45, attivabile dalla barra principale degli strumenti.
Se, come in Figura 45, è attivato l'interruttore dell'autocomposizione (quello con l'icona della bacchetta magica), a seguito dell'inserimento di uno qualsiasi degli oggetti partirà in automatico un'autocomposizione che ne faciliterà l'impostazione delle proprietà.

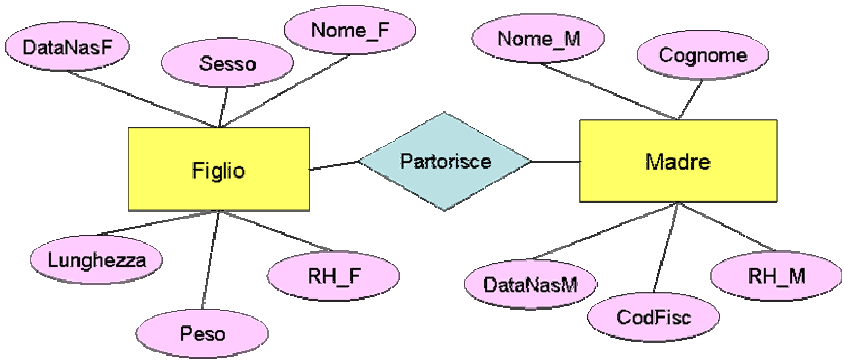
Figura 45: La casella degli strumenti ActiveX di Microsoft Access®

Per concludere, va ricordato che ogni oggetto della maschera dispone di una serie di parametri di configurazione, denominati proprietà, a cui si accede mediante il comando Proprietà posto nella barra principale degli strumenti.

6 Soluzione degli esercizi

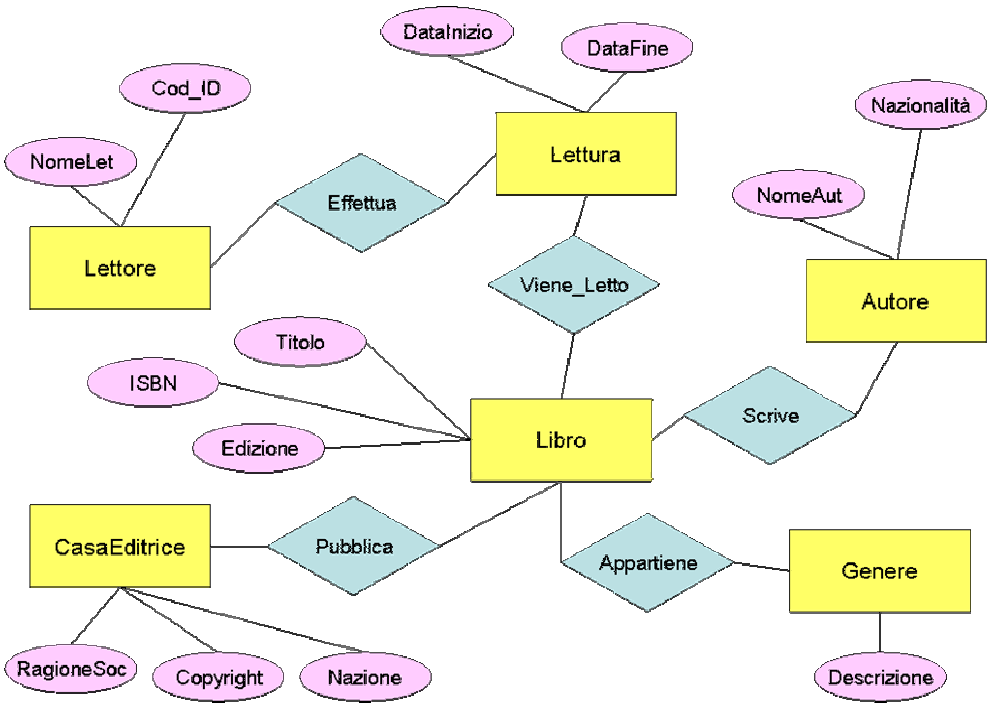
Esercizio 1 (capitolo 2.3.6)

Proponiamo il modello E/R seguente. Si noti che per gli attributi concettualmente simili delle due entità Madre e Figlio (DataNascita, Nome, RH) sono stati usati nomi leggermente differenti, onde evitare ambiguità nelle successive fasi di modellizzazione.



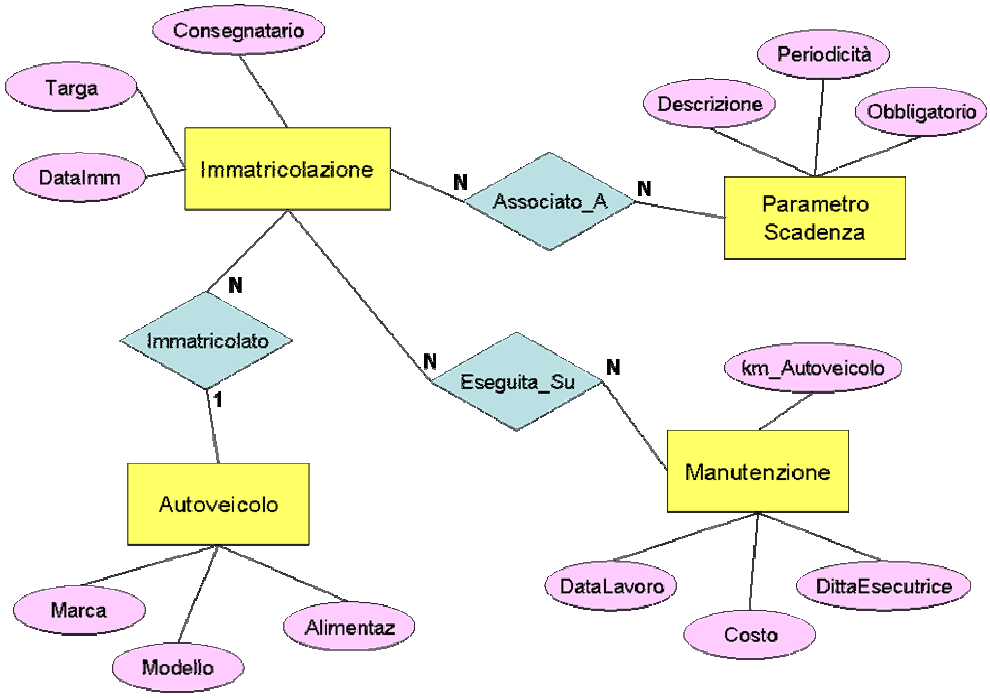
Esercizio 2 (capitolo 2.3.6)

Proponiamo il modello E/R seguente. Si osservi che l'entità Lettura è di tipo concettuale, mentre tutte le altre sono di tipo fisico.



Esercizio 3 (capitolo 2.3.6)

Proponiamo il modello E/R seguente. Riteniamo che la (presunta) difficoltà legata alla realizzazione del modello sia dovuta principalmente alla presenza di molte entità di tipo concettuale (Immatricolazione, ParametroScadenza, Manutenzione), le quali possono risultare "meno visibili" delle corrispondenti entità fisiche (Autoveicolo). Inoltre la descrizione formale del caso studio appare oggettivamente poco dettagliata; in conseguenza a ciò possono esserci differenti interpretazioni da parte del lettore che conseguentemente possono portare a modelli E/R diversi da quello proposto (e comunque tutti sostanzialmente corretti). E' chiaro infatti che la scarsità di informazioni di input non può che comportare un allargamento delle posizioni interpretative da parte del modellista, che invece si restringono nel caso di un'impostazione precisa e dettagliata dei requisiti iniziali. Si ribadisce quindi l'estrema importanza della fase di analisi iniziale descritta al capitolo 2.2.



Esercizio 4 (capitolo 2.4.12)

Sicuramente anche l'attributo NumeroTelaio può essere considerato PK. Anzi, il NumeroTelaio è proprio quel parametro che, anche da un punto di vista giuridico, identifica univocamente e per sempre un autoveicolo. Viceversa il numero di targa può cambiare nel tempo, per esempio in caso di una nuova immatricolazione.

La questione deve quindi essere affrontata sotto il profilo semantico: per Autoveicolo si intende l'oggetto fisico dotato di carrozzeria, motore, ruote e tutto il resto (quindi sostanzialmente come oggetto immutabile), oppure si pensa ad un'entità concettuale legata all'evento "immatricolazione" che, in quanto tale, può cambiare nel tempo?
Entrambi gli approcci sono corretti. Basta capirsi prima che sia troppo tardi.
A tale scopo si osservi il modello E/R proposto nell'esercizio 3. In quel caso l'Autoveicolo è stato interpretato come un'entità legata all'evento Immatricolazione e, volutamente, non è stato utilizzato l'attributo NumeroTelaio onde evitare ridondanza (nel libretto di immatricolazione, al numero di targa è associato sempre il numero di telaio, quindi il secondo deriva dal primo).

Esercizio 5 (capitolo 2.4.12)

Certo che sì. Infatti CF è FK in VisitaMedica, quindi in quanto tale può ripetersi, il che equivale ad ammettere che uno stesso paziente (CF) può benissimo essere sottoposto a molte visite mediche.

Esercizio 6 (capitolo 4.3.3)

Per maggiore chiarezza riportiamo il testo di Interrogazione 2 integrato dal nuovo dato richiesto nel presente esercizio (in corsivo):

> **Interrogazione 2a**
> Estrarre i pazienti che hanno effettuato visite mediche nel reparto di ortopedia, *indicando il nominativo del medico che effettua le visite.*

Si tratta naturalmente di realizzare una query di selezione in cui devono essere inserite le tabelle VisitaMedica, Paziente, Reparto e PersonaleMedico, avendo cura di impostare il filtro corrispondente al Reparto di ortopedia (CodRep "OTR4"). Il risultato è visualizzato in Figura 46.
Provando ad eseguire la query così impostata, ci si accorge però che essa non restituisce alcun risultato, pur essendoci nel database un'istanza che soddisfa i criteri impostati (controllare in Figura 28). Perché accade questo?
In Figura 46 si osservino i JOIN che collegano le tabelle VisitaMedica a Reparto e VisitaMedica a PersonaleMedico. Avendo in mente la definizione algebrica del JOIN, si capisce subito che l'istanza risultante dalla query siffatta deve collegare

contemporaneamente le istanze di VisitaMedica con le istanze di Reparto e PersonaleMedico. Se però, come nell'esempio, capita che un medico sia afferente ad un reparto diverso da quello dove va ad effettuare la visita medica (infatti il medico pediatra Rossi Paolo - Matr. 234G567 che fa la visita nel reparto di Ortopedia - CodRep ORT4 è afferente al reparto di Pediatria - CodRep PDT04), il doppio JOIN restituisce un'istanza nulla, in quanto *"nelle istanze non c'è alcun medico che afferisca al reparto di Ortopedia e che faccia anche una visita medica nello stesso reparto"*.

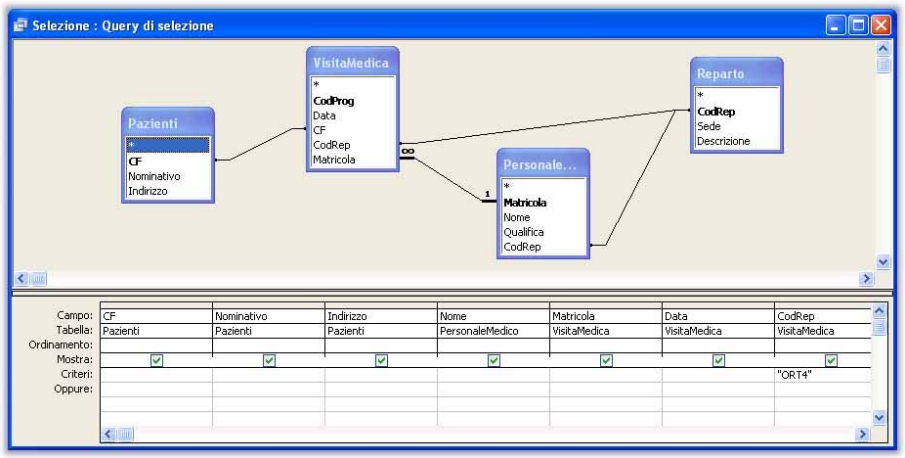


Figura 46: "Prima versione" della struttura di query di selezione per l'estrazione dei dati richiesti

La query di Figura 46 deve essere allora parzialmente modificata mediante la cancellazione del JOIN indesiderato, ovvero quello che collega Reparto con PersonaleMedico (basta selezionare il simbolo di JOIN nella struttura di query e premere Canc), come mostrato in Figura 47. E' importante sottolineare che la modifica (eliminazione o aggiunta) di un JOIN nella struttura di query non comporta alcuna conseguenza sulla struttura logica del database.

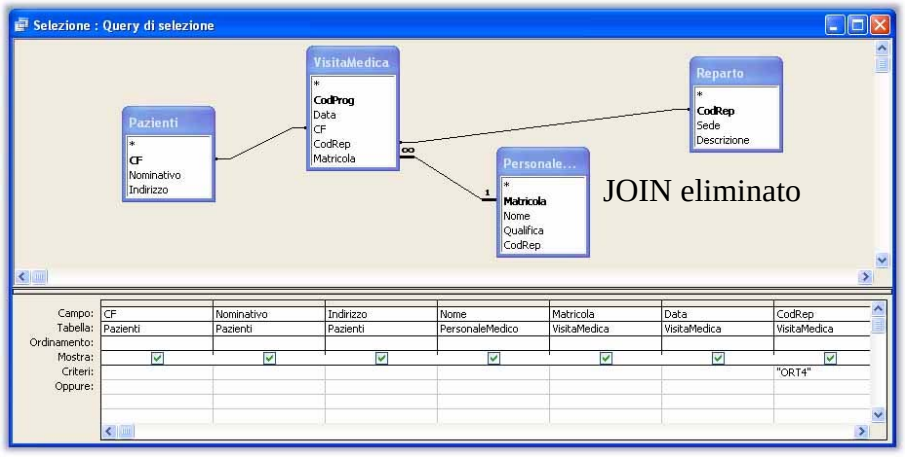


Figura 47: "Versione corretta" della struttura di query di selezione per l'estrazione dei dati richiesti

Esercizio 7 (capitolo 4.3.3)

Per maggiore chiarezza riportiamo il testo di Interrogazione 2 integrato dal nuovo dato richiesto nel presente esercizio (in corsivo):

> **Interrogazione 2b**
>
> Estrarre i pazienti che hanno effettuato visite mediche nel reparto di ortopedia, *visualizzando tutti i medici (a prescindere che abbiano fatto o meno visite mediche) e, in riferimento alle visite mediche effettuate, riportare i campi Data, Paziente e Reparto.*

Con questo esercizio si incontra (per la prima volta in questa dispensa) il concetto di valore *Null* (vuoto). Trattasi di un concetto importante, usato per rappresentare i valori di attributi sconosciuti, o non applicabili perché inesistenti, per una certa tupla. Anche in riferimento all'esercizio 6, si vede infatti che non tutti i medici hanno eseguito visite registrate nel database (nella fattispecie il solo medico ad aver fatto visite è Rossi Paolo - Matr. 234G567).

Quindi ci si aspetta che la query cercata realizzi, oltre alle associazioni tra il medico suddetto e le relative visite da lui effettuate unitamente agli altri dati richiesti, anche le associazioni tra gli altri medici (quelli che non hanno fatto visite) e ... una serie di campi vuoti (null) non essendoci appunto visite mediche a loro associate.

Procediamo per gradi.

Prima di tutto realizziamo una query di selezione in modalità struttura analoga a quella già fatta nel precedente esercizio 6 avendo cura di cancellare il JOIN non necessario e di inserire tutti i campi richiesti dall'interrogazione modificata. Il risultato finale è visibile in Figura 48.

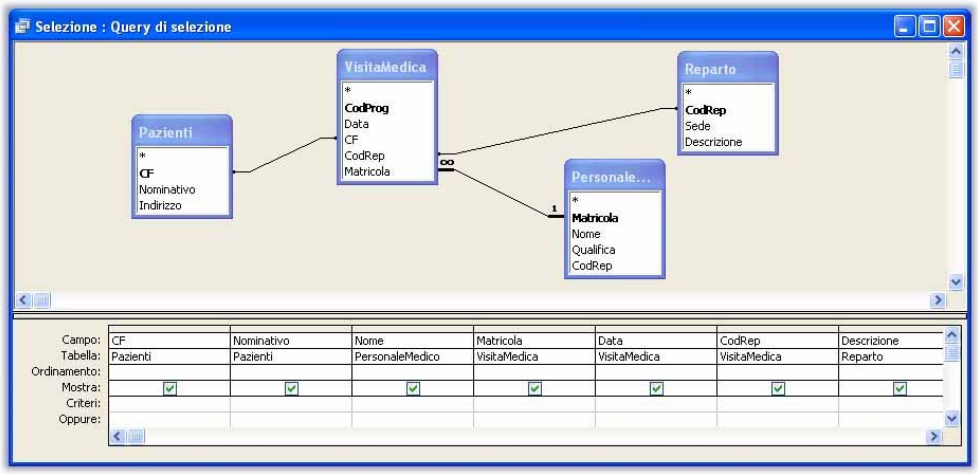


Figura 48: Prima fase dell'impostazione della query per realizzare i requisiti di Interrogazione 2b

Successivamente, come suggerito nel testo dell'esercizio a pagina 60, proviamo a fare doppio click sul simbolo di JOIN tra PersonaleMedico e VisitaMedica. Compare l'interfaccia di Figura 49, nella quale bisogna impostare l'opzione 2 che specifica di includere tutti i record di PersonaleMedico (quindi anche i medici che non hanno fatto visite mediche) e solo i record di VisitaMedica in cui i campi calcolati (PK=FK) sono uguali. La seconda parte della specifica, apparentemente inutile ma i realtà fondamentale, serve per stabilire che non deve essere fatto un prodotto cartesiano, bensì appunto un JOIN.

Figura 49: Interfaccia per l'impostazione delle proprietà JOIN tra le tabelle PersonaleMedico e VisitaMedica. Notare che, per rispondere all'Interrogazione 2b, è stata selezionata l'opzione 2

Purtroppo non è finita, perché se dopo questa impostazione si prova ad eseguire nuovamente la query ci si imbatte nell'inquietante messaggio di Figura 50.

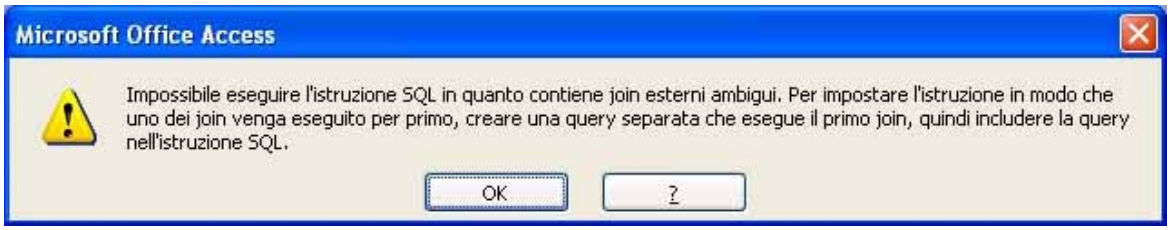


Figura 50: Messaggio di errore derivante dall'impostazione di JOIN ambigui

Vediamo di capire meglio cosa sta succedendo. Esaminando la struttura di query di Figura 48 e considerate le nuove impostazioni del JOIN di Figura 49, si deve dedurre che al DBMS sono state imposte le seguenti tre regole algebriche, che devono valere contemporaneamente:

1. Considera tutti i campi inseriti nella query di Figura 48
2. Mostra tutti i record di PersonaleMedico
3. Mostra solo i record di VisiteMediche che trovano corrispondenza con PersonaleMedico, Reparto e Paziente

Un esame approfondito delle tre regole consente abbastanza facilmente di evidenziare che le regole n. 2 e n. 3 sono in contraddizione tra loro: mentre la n. 2 pretende di visualizzare tutti i medici, la n. 3 impone che siano estratte solo le visite mediche fatte (ovviamente dai medici). Ecco il motivo del messaggio di errore di Figura 50.
Bisogna allora modificare anche gli altri JOIN della struttura secondo gli schemi riportati in Figura 51.

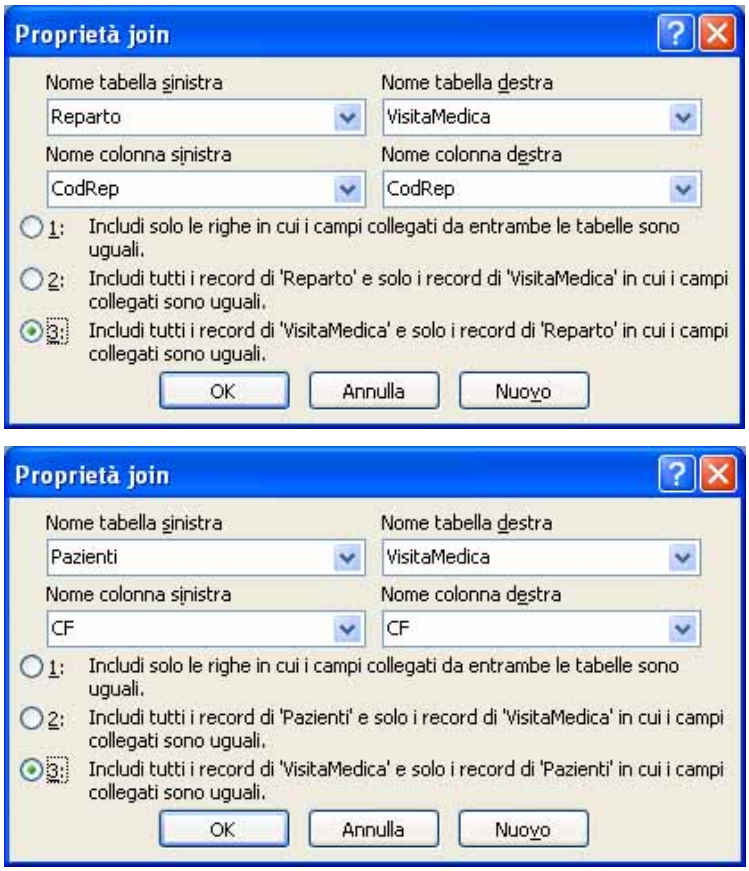


Figura 51: Interfaccia per l'impostazione delle proprietà JOIN tra le tabelle Reparto→VisitaMedica (opzione 3) e Paziente→VisitaMedica (opzione 3)

Dopo le citate modifiche la simbologia grafica della struttura di query cambia leggermente, visualizzando una freccia "verso" la tabella (1) dell'associazione 1→N definita dal JOIN, come mostrato in Figura 52.
Convenzionalmente un JOIN che visualizza tutte le istanze della tabella (1) è definito RIGHT JOIN, mentre uno che visualizza tutte le istanze della tabella (N) è definito LEFT JOIN. In Figura 53 è visualizzata la stringa SQL della query, nella quale sono individuabili le clausole LEFT JOIN e RIGHT JOIN ed infine, in Figura 54, è visualizzato il risultato della query. Si notino i valori *null* (caselle vuote)

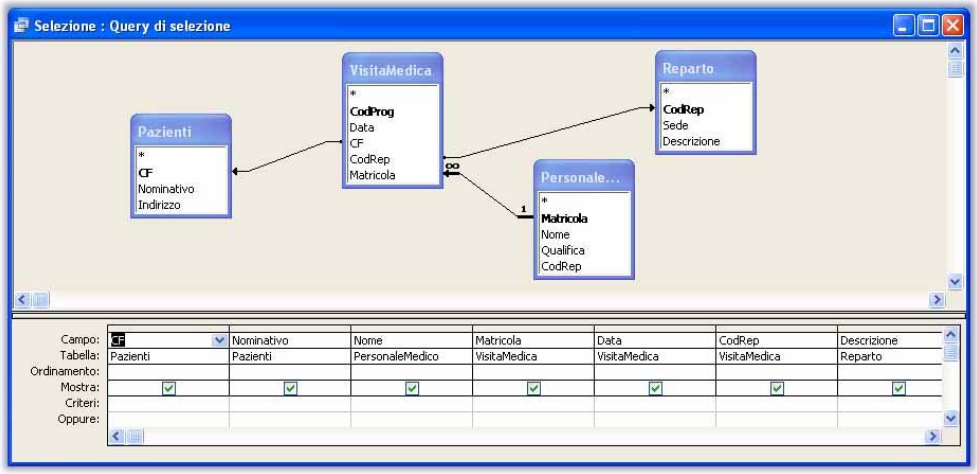


Figura 52: La strruttura definitiva della query per realizzare i requisiti di Interrogazione 2b, in cui sono visibili le simbologia grafiche degli operatori LEFT JOIN e RIGHT JOIN.

Figura 53: Stringa SQL corrispondente alla query definitiva di Figura 52.

Figura 54: Risultato della query definitiva di Figura 52. Si notino i campi vuoti (null) in corrispondenza dei medici che non hanno eseguito alcuna visita.

Esercizio 8 (capitolo 4.3.3)

La scelta del campo da utilizzare per il conteggio delle istanze deve essere fatta tra quelli che sicuramente, in ogni istanza, contengono un valore $\neq null$. Tutto ciò al fine di ottenere effettivamente il reale numero di istanze registrate. Nel caso di Interrogazione 3, in riferimento agli esempi di Figura 28, anche il campo Data soddisfa il requisito appena formulato; tuttavia, almeno potenzialmente e a prescindere dal fatto che negli esempi trattati ciò sia stato fatto o meno, il campo Data potrebbe contenere un valore *null* (basta "dimenticarsi di inserire la data della visita medica"). Questo non può però accadere per il campo CodProg in quanto esso è PK ed il DBMS non consente di inserire istanze in cui il campo PK sia *null*. Quindi la regola che è bene non dimenticare prevede di utilizzare sempre un campo PK per la formula *conteggio*.

7 Bibliografia selezionata e indirizzi web

Bibliografia selezionata

American National Standards Institute: *"The Database Language SQL"*, Document ANSIX3.135, 1986.

Chen, P.P.: *"The Entity Relationship Mode - Torward a Unified View of Data"*, TODS, Vol. 1, Numero 1, (03/1976).

Codd, E.F.: *"A Relational Model for Large Shared Data Banks"*, CACM, Volume 13, Numero 6, giugno 1970.

Elmasri, R.A., Navathe, S.B.: *"Sistemi di basi di dati. Fondamenti."* IV Edizione, Ed. Pearson/Addison Wesley, settembre 2004.

Prague, C.N., Irwin, M.R.: *"Microsoft Access 2000"*, (IDG Books), Tecniche nuove, 1999.

Indirizzi web suddivisi per argomento principale (gli indirizzi riportati risultano attivi al 04/01/2006)

Algebra relazionale

http://www.dis.uniroma1.it/~catarci/Cap3.pdf
http://it.wikipedia.org/wiki/Algebra_relazionale

Microsoft Access

http://office.microsoft.com/it-it/FX010857911040.aspx
http://www.mclink.it/personal/MC5884/

Modellistica

http://bit.csc.lsu.edu/~chen/chen.html
http://www.itworld.com/nl/db_mgr/05072001/pf_index.html
http://en.wikipedia.org/wiki/Codd%27s_12_rules

SQL

http://www.html.it/sql/
http://www.sqlcourse.com/
http://www.w3schools.com/sql/default.asp

Visual Basic for Application

http://msdn.microsoft.com/isv/technology/vba/default.aspx
http://www.itportal.it/tutorial/software/office/vba/
http://msdn.microsoft.com/vbasic/