Carlo Niccolai (carlopavana)

DBMS - I DATABASE RELAZIONALI

28 April 2022

Un poco di teoria

I sistemi di database relazionali sono diventati sempre piu' popolari a partire dalla fine degli anni '70 ed offrono un metodo potente per l'archiviazione dei dati in modo indipendente dalle applicazioni e per molti utilizzatori la banca dati e' il fulcro della strategia dell'Information Tecnology, dato che attorno ad una struttura di database un sistema puo' svilupparsi in modo stabile, sicuro, affidabile, efficiente, e trasparente.
Nei primi sistemi, intorno agli anni '60, ogni gruppo di programmi applicativi, ogni procedura, aveva la sua propria struttura di file indipendente, ma la inevitabile duplicazione dei dati, assai frequentemente, portava a risultati aziendali incoerenti e naturalmente molti sforzi sono stati fatti per costruire delle strutture di dati che potessero essere comuni a molti programmi applicativi e a molti sistemi di sviluppo.
Tecniche di strutturazione dei dati, sviluppate per sfruttare la memorizzazione ad accesso casuale dei dispositivi, i dischi magnetici, hanno aumentato la complessita' delle operazioni sui dati, per inserirli, eliminarli, aggiornarli e consultarli .
Come primo passo verso un DBMS furono introdotti dei pacchetti di subroutine software per ridurre lo sforzo del programmatore per mantenere e condividere queste strutture di dati.
Tuttavia, l'uso di questi pacchetti richiedeva ancora la conoscenza della organizzazione fisica dei dati.

L'approccio al database

Un sistema di database e' basato su computer per la memorizzazione e la conservazione di informazioni, le informazioni in questione possono essere qualcosa di significativo per l'organizzazione per il cui utilizzo sono destinati.
Un database puo' contenere una varieta' di cose diverse. per rendere la progettazione dei database piu' duratura nel tempo, i contenuti dei database sono divisi in due concetti:

- Schema
- Dati

Lo schema e' la struttura dei dati, mentre i dati sono i "fatti". All'inizio lo schema puo' essere complesso da capire, ma in realta' indica le regole a cui i dati, nel loro sviluppo, devono obbedire.
Immaginate un caso in cui si desideri memorizzare i dati sui componenti presenti in un laboratorio amatoriale, tali fatti potrebbero includere la tipologia, il codice, la descrizione e la collocazione in un magazzino locale o remoto.

In un database tutte le informazioni su tutti i componenti si terra' in un unico stoccaggio "contenitore", chiamato **tabella**. Questa tabella e' un oggetto tabellare come una pagina di foglio di calcolo, con i diversi componenti come le righe, e i fatti (per esempio i loro codici) come colonne. Chiamo questa tabella CREMA, e gli elementi potrebbero essere simili a:

Classe	Tipo	Codice	Descrizione	Quantita'	Contenitore
SE	TRANSISTOR	BC461	Small Signal Transistor General Purpose	3	B03
SE	DIODO	1N4007	General Purpose Plastic Rectifier	10	C09
AN	FERROCUBE	AF01	Antenna in ferrocube con bobina di sintonia	2	A02
TR	ALI	TRALI-08	Trasformatore di alimentazione 200VA	1	T01
TR	ALI	TRALI-10	Trasformatore di alimentazione 100VA	2	T01

Da queste informazioni lo schema che definisce CREMA ha sei componenti, Classe, Tipo, Codice, Descrizione, Quantita', Contenitore.
Durante la progettazione si possono chiamare le colonne come piu' ci piace, ma aiuta se sono usati nomi significativi.
Oltre al nome, vogliamo cercare di fare in modo che gli utenti non inseriscano accidentalmente un nome nella colonna QUANTITA', o commettano qualche altro stupido errore. Proteggere il database rispetto a dati spazzatura e' uno dei passi piu' importanti nella fase di progettazione del database.
Da quello che sappiamo sui dati (i fatti), possiamo dare delle regole e queste definizioni:

- CLASSE e' il raggruppamento maggiore dei componenti da classificare. Non duplicabile, e' di due caratteri. <li? TIPO Identifica il componente nell'ambito della classe. Sono 10 caratteri e non e' duplicabile nella classe.
- CODICE E' il codice del componente, quello assegnato dal produttore o uno attribuito per identificarlo. Non e' duplicabile nell'ambito della classe/tipo.
- DESCRIZIONE: Descrizione del componente.Le parole piu' lunghe di tre caratteri entrano nella ricerca alfabetica.
- QUANTITA': Un numero se zero non e' piu' presente, se -1 sono molti e non contati.
- CONTENITORE: E' il codice del contenitore ove e' conservato questo componente. Lo stesso contenitore puo' conservare piu' componenti diversi.

Questo e' un modello semplificato di CREMA, serve per la discussione sui database. Lo schema completo e dettagliato e' contenuto in altro documento.
Durante la fase di progettazione di uno schema di database le regole, simili a quelle illustrate o anche piu' complesse sono definite e ove possibile implementate. Piu' le norme sono rigide e piu' e' difficile caricare e ritrovarsi nel tempo dati di scarsa qualita'.

Tipi di utenza

Quando si considerano gli utenti di un sistema di database, ci sono tre grandi categorie da considerare:

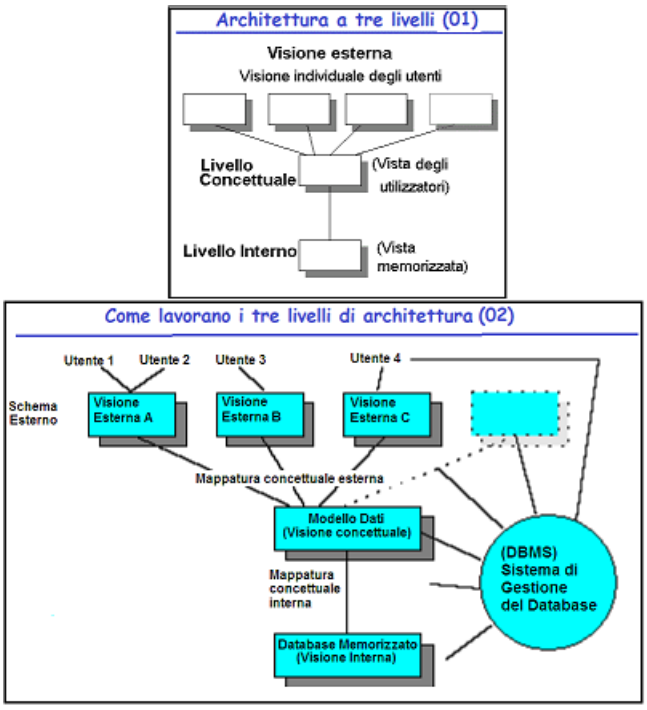
1. Il programmatore delle applicazioni, responsabile per la scrittura di programmi con alcuni linguaggi di alto livello come COBOL, C + +, ecc
2. L'utente finale, che accede al database attraverso un linguaggio di interrogazione.
3. L'amministratore del database (DBA), che controlla tutte le operazioni sul database, la coerenza delle regole e la loro aderenza alle problematiche aziendali.

Architettura del Database

Non tutti i DBMS sono conformi alla stessa architettura.

- L'architettura a tre livelli costituisce la base della struttura delle banche dati moderne.
- Questo e' in accordo con il gruppo di studio l'ANSI/SPARC sulla gestione dei sistemi di database
- ANSI/SPARC e' l'American National Standards Institute/Standard Planning and Requirement Committee.
- L'architettura dei DBMS e' divisa in tre livelli generali:
    - Esterno
    - Concettuale
    - Interno

Architettura del database a tre livelli.



ey01.png

1. Il livello esterno: riguarda le modalita' con cui gli utenti possano visualizzare i dati
2. Il livello concettuale: puo' essere considerato come la vista di una comunita' di utenti; una descrizione formale dei dati di interesse per l'organizzazione, indipendentemente da qualsiasi considerazione di archiviazione.
3. Il livello interno: riguarda le modalita' in cui i dati sono effettivamente memorizzati.

Visione esterna

Un utente e' chiunque abbia bisogno di accedere a una porzione dei dati. Essi vanno da programmatori di applicazioni a utenti occasionali con interrogazioni ad hoc. Ogni utente dispone di un linguaggio a sua disposizione.
Il programmatore puo' usare un linguaggio di alto livello (es. COBOL), mentre l'utente occasionale probabilmente usa un linguaggio di interrogazione (query). Mbr>

Indipendentemente dal linguaggio usato, includera' per i dati un sottolinguaggio DSL(Domain Specific Language) che e' quel sottoinsieme del linguaggio che si occupa della memorizzazione e del reperimento delle informazioni nel database e puo' o non puo' essere evidente all'utente.
Un DSL e' una combinazione di due linguaggi:

- Un Data Definition Language (DDL) - che fornisce la definizione o una descrizione di oggetti deli database

- Un linguaggio di manipolazione dei dati (DML) che supporta la manipolazione o la elaborazione degli oggetti di database.

Ogni utente vede i dati in termini di visione esterna: Definita da uno schema esterno, composta essenzialmente dalla descrizioni di ciascuno dei vari tipi di record esterno in quella vista esterna, e anche una definizione della mappatura tra lo schema esterno e il sottostante schema concettuale.

Vista concettuale

- Una rappresentazione astratta dell'intero contenuto informativo del database.
- Si tratta, in generale, di una vista dei dati come sono realmente, cioe', si tratta di un modello del del mondo reale
- Si compone di piu' occorrenze di molteplici tipi di record definiti nello schema concettuale.
- Per raggiungere l'indipendenza dei dati, la definizione di record concettuale deve coinvolgere solo il contenuto informativo.
- La struttura di memorizzazione viene ignorata
- La strategia di accesso viene ignorata.
- In aggiunta alle definizioni, lo schema concettuale contiene le autorizzazioni e le procedure di convalida.

Visione interna

Il punto di vista interno e' un livello basso di rappresentazione dell'intero database composto da piu' occorrenze di molteplici tipi di record interni memorizzati.
E' comunque ad un passo dal livello fisico dato che non si occupa in termini di record fisici o blocchi ne' con i vincoli dei dispositivi come il cilindro o le dimensioni della pista. I dettagli della mappatura della memorizzazione fisica sono altamente specifici nella implementazione e non sono espressi nei tre livelli architettura.
Il punto di vista interno come descritto dallo schema interno:

- Definisce i vari tipi di record memorizzati
- Quali sono gli indici esistenti
- Come sono rappresentati campi memorizzati.
- In quale sequenza fisica sono i record memorizzati.

In effetti lo schema interno e' la definizione della struttura di memorizzazione.

Mappatura

- La mappatura concettuale / interna:
    - Definisce la corrispondenza tra vista concettuale e vista interna
    - Precisa la mappatura da record concettuale alla loro controparte memorizzata.
- Una mappatura esterna/concettuale:

    - Definisce una corrispondenza particolare tra vista esterna e concettuale
- Una modifica alla definizione della struttura di memorizzazione significa che la mappatura interna/concettuale deve essere modificata di conseguenza, in modo che lo schema concettuale possa rimanere invariante raggiungendo l'indipendenza fisica dei dati.
- Una modifica della definizione concettuale significa che la mappatura esterna/ concettuale deve essere modificata di conseguenza, in modo che lo schema esterno possa rimanere invariante, raggiungendo l'indipendenza logica dai dati.

DBMS

Il database management system (DBMS) e' il software che:

- Gestisce tutti gli accessi alla banca dati
- E' responsabile per applicare i controlli di autorizzazione e le procedure di validazione

Concettualmente cio' che accade e' questo:

1. Un utente invia una richiesta di accesso, utilizzando alcuni particolari DML.
2. Il DBMS intercetta le richieste e le interpreta.
3. Il DBMS ispeziona a sua volta lo schema esterno, la mappatura concettuale/esterna, lo schema concettuale, la mappatura concettuale interna e la memorizzazione delle strutture di definizione.
4. Il DBMS esegue le operazioni necessarie sul database memorizzato.

Database Administrator

L'amministratore del database (DBA) e' la persona (o gruppo di persone) responsabile per il controllo complessivo del sistema di database. Le responsabilita' del DBA includono quanto segue:

- Decidere il contenuto informativo del database, cioe' individuare le entita' di interesse per l'impresa e le informazioni da registrare su quelle entita'. Questo e' definito dalla scrittura dello schema concettuale utilizzando il DDL
- Decidere la struttura di memorizzazione e la strategia di accesso, vale a dire come i dati devono essere rappresentati dalla scrittura alla definizione della struttura di memorizzazione. Anche lo schema associato concettuale/interno deve essere specificata utilizzando il DDL.
- Contattare gli utenti, cioe' garantire che i dati di cui hanno bisogno siano disponibili e scrivere gli schemi esterni necessari e le mappature esterne/concettuali (ancora una volta usando il DDL)
- Definendo i controlli di autorizzazione e le procedure di validazione. I controlli di autorizzazione e le procedure di validazione sono estensioni allo schema concettuale e possono essere specificati con il DDL

- Definire una strategia di backup e ripristino. Per esempio lo scarico periodica del database su una memoria di backup e le procedure per la ricarica del database per sicurezza. L'uso di un file di log in cui ogni record di log contiene i valori degli elementi del database prima e dopo un cambio e puo' essere utilizzato per scopi di recupero dei dati.
- Monitorare le prestazioni e rispondere ai cambiamenti nei requisiti, cioe' cambiando i dettagli di memorizzazione e di accesso cosi' organizzando il sistema in modo da ottenere le prestazioni che siano il meglio per la gestione.

Strutture e limitazioni

I servizi offerti dal DBMS possono variare molto a seconda del loro livello di sofisticazione. Comunque, in generale, un buon DBMS deve fornire questi vantaggi rispetto a un sistema convenzionale:

- L'indipendenza dei dati dal programma - Si tratta del vantaggio principale di un database. Sia il database che il programma utente possa essere modificati indipendentemente l'uno dall'altro risparmiando cosi' tempo e denaro che sarebbero necessari per mantenere la consistenza.
- Condivisibilita' dei dati e non ridondanza dei dati. La situazione ideale e' quella di consentire alle applicazioni di condividere un database integrato contenente tutti i dati necessari per le applicazioni ed eliminare cosi' il piu' possibile la necessita' di memorizzare i dati in modo ridondante.
- Integrita' - Con molti utenti diversi e la condivisione di varie parti del database, non e' possibile per ogni utente essere responsabile della consistenza dei valori nel database o per mantenere le relazioni tra gli oggetti o dati utente e tutti gli altri elementi, alcuni dei quali possono essere sconosciuti o addirittura proibiti all'utente.
- Controllo centralizzato - Con il controllo centrale del database, il DBA puo' garantire che gli standard siano seguiti nella rappresentazione dei dati.
- Sicurezza - Avere il controllo sulla banca dati dal DBA puo' garantire che l'accesso al database avvenga attraverso i canali appropriati e sia possibile definire i diritti di accesso di qualsiasi utente di qualsiasi elemento o un sottoinsieme di dati definiti del database.
- Il sistema di sicurezza deve impedire la corruzione dei dati esistenti sia accidentalmente che intenzionalmente.
- Prestazioni ed efficienza - In considerazione delle dimensioni del database e delle richieste di accesso al database, sono requisiti importanti buone prestazioni ed efficienza. Conoscendo le esigenze globali della organizzazione, a differenza alle esigenze di ogni singolo utente, il DBA puo' strutturare il sistema database per fornire un servizio completo che sia **migliore per la gestione**

Indipendenza dei dati

- Questo e' un vantaggio principale di un database. Sia il database che il programma utente possono essere modificati in modo indipendente l'uno dall'altro.
- I sistemi di applicazione convenzionali sono dati-dipendenti. Cio' significa che il modo in cui sono organizzati i dati nella memoria secondaria e il modo in cui si accede sono entrambe dettate dalle esigenze dell'applicazione, e, inoltre, la conoscenza della organizzazione dei dati e la tecnica di accesso e' integrato nella logica dell'applicazione.
- Per esempio, se un file viene memorizzato in forma sequenziale indicizzato allora l'applicazione deve conoscere
    - Quale indice esiste
    - L'ordine del file (come definito dall'indice)

  La struttura interna dell'applicazione sara' imperniata intorno a questa conoscenza. Se, ad esempio, il file e' stato sostituito da un file diversamente indicizzato, maggiori modifiche dovrebbero essere apportate all'applicazione.
  Tale applicazione e' dipendente dai dati - non e' possibile cambiare la struttura di archiviazione (come i dati vengono fisicamente registrati) o la strategia di accesso (come si accede) senza alterare, probabilmente drasticamente, l'applicazione. Le parti da cambiare nell'applicazione sono quelle che comunicano con il software di gestione dei file - le difficolta' sono abbastanza irrilevanti per il problema se l'applicazione e' stata scritta per risolverlo.

    - Non e' opportuno consentire alle applicazioni di essere dipendente dai dati diverse applicazioni avranno bisogno di diversi punti di vista degli stessi dati.
    - L'amministratore deve avere la liberta' di cambiare la struttura di memorizzazione o la strategia di accesso in risposta alle mutevoli esigenze, senza dover modificare le applicazioni esistenti.
    - L'indipendenza dei dati puo' essere definito come Applicazioni immuni dai cambiamento nella struttura, nella memorizzazione e nella strategia di accesso.

La ridondanza dei dati

In un sistema di non-database ogni applicazione ha i suoi file privati. Questo spesso puo' portare a ridondanza nei dati memorizzati, con risultante perdita di

spazio di archiviazione. In un database i dati sono tra loro integrati.
Il database puo' essere pensato come una unificazione di diversi file distinti di dati, con qualsiasi ridondanza tra i file parzialmente o completamente eliminta.
L'integrazione dei dati e' generalmente considerata come una caratteristica importante di un database. L'eliminazione della ridondanza dovrebbe essere un obiettivo, tuttavia, il vigore con cui questo obiettivo dovrebbe essere perseguito e' una questione ancora aperta.
La ridondanza e'

- Diretta se un valore e' una copia di un altro
- Indiretta, se il valore puo' essere derivato da altri valori:
  - Semplifica la ricerca ma complica l'aggiornamento
  - Al contrario l'integrazione rende il recupero piu' lento e gli aggiornamenti piu' facili
- La ridondanza dei dati puo' portare ad inconsistenze nel database a meno di notevoli controlli.
- Il sistema dovrebbe essere a conoscenza di eventuali duplicazione dei dati - il sistema e' responsabile che gli aggiornamenti siano eseguiti correttamente.
- Un DB con ridondanza non controllata puo' essere in uno stato incoerente - puo' fornire informazioni inesatte o in conflitto
- Un dato di fatto rappresentato da una singola voce non puo' portare a inconsistenza - pochi sono i sistemi grado di propagare gli aggiornamenti cioe' la maggior parte dei sistemi non supportano la ridondanza controllata.

L'integrita' dei dati

Un DBMS deve garantire che i dati nel database siano accurati

- L'incoerenza tra due voci che rappresentano lo stesso "fatto" danno un esempio di mancanza di integrita' (causata dalla ridondanza nel database).
- Il vincolo di integrita' puo' essere visto come un insieme di linee guida da seguire quando l'aggiornamento di un DB deve essere conservato senza errori.
- Anche se la ridondanza viene eliminata, il DB potrebbe ancora contenere dati non corretti.
- I controlli di integrita' che sono importanti sono controlli sugli elementi dei dati e sui tipi di record.

I controlli di integrita' sui dati possono essere divisi in 4 gruppi:

1. Controllo della tipologia

- Esempio garantire che un campo numerico lo sia effettivamente - questo controllo deve essere eseguita automaticamente dal DBMS.

2. Controllo di ridondanza
    - Ridondanza diretta o indiretta (vedi la ridondanza dei dati) - questo controllo nella maggior parte dei casi non e' automatico.
3. Controllo di gamma
    - Esempio per garantire che un dato valore rientri in un intervallo specifico di valori, come controllare le date in modo che l'eta' sia > 0 o < 110.
4. Controlli di confronto
    - Con questo controllo un insieme di valori dato e' confrontato con i valori di un altro insieme di valori. Per esempio, lo stipendio massimo per un dato insieme di dipendenti deve essere inferiore al salario minimo per l'insieme di dipendenti di stipendio piu' alto.

Un tipo di record puo' avere vincoli sul numero totale di presenze, o di inserimenti e cancellazioni di record. Per esempio in un database dei pazienti potrebbe esserci un limite al numero di esami per raggi x per ogni paziente o i dettagli di una visita in ospedale i pazienti devono essere conservati per un minimo di 5 anni prima che possano essere cancellati

- Il controllo centralizzato del database aiuta a mantenere l'integrita' e permette al DBA di definire le procedure di validazione da effettuare ogni volta che sia tentata una operazione di aggiornamento (l'aggiornamento copre la modifica, la creazione e la cancellazione).
- L'integrita' e' importante in un sistema di database un'applicazione eseguita senza procedure di validazione in grado di produrre dati errati che possono poi influenzare altre applicazioni che utilizzano tali dati.

Analisi del Database

In questa sezione mi occupo del processo di assunzione delle specifiche della banca dati da un cliente e della implementazione della sottostante struttura del database necessaria per supportare le sue indicazioni. E' quello che spesso viene chiamato processo di analisi.

Introduzione

L'analisi dei dati si occupa della natura e dall'uso dei dati. Si tratta di identificare gli elementi dei dati necessari per supportare il sistema di elaborazione dei

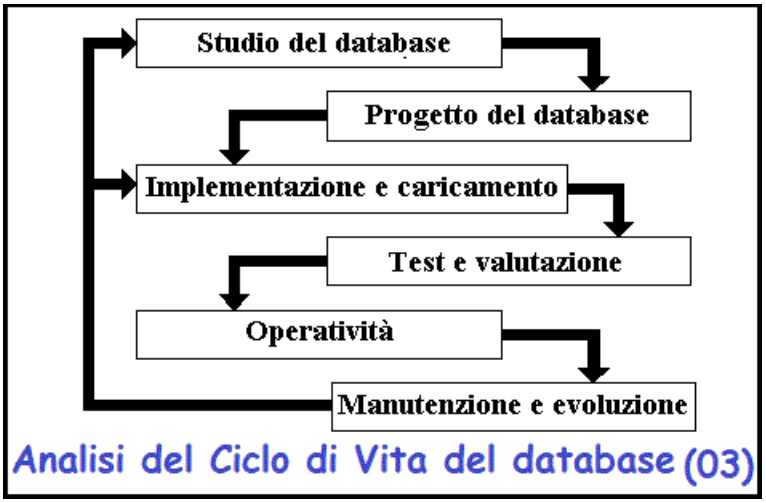
dati dell'organizzazione, la collocazione di questi elementi in gruppi logici e la definizione dei rapporti tra i risultanti gruppi.
Sono possibili altri approcci, per esempio D.F.D. e diagrammi di flusso, ma sono metodologie che esulano da questa breve trattazione.
Nella pratica spesso l'analisi del sistema passa direttamente dalla riceca dei fatti alla realizzazione della dipendente analisi dei dati. Le ipotesi di utilizzo delle proprieta' e delle relazioni tra gli elementi dei dati sono incorporate direttamente nel progetto dei file e dei record e nelle specifiche della procedura informatica.
L'introduzione dei Database Management Systems (DBMS) ha favorito un maggiore livello di analisi, in cui sono definiti gli elementi dei dati che provengono da un modello logico o 'schema' (schema concettuale).
Quando si discute lo schema nel contesto di un DBMS, si considera l'effetto di progetti alternativi sulla efficienza o la facilita' di implementazione, cioe' l'analisi e' ancora dipendente dall'implementazione. Se si considerano le relazioni tra i dati, gli usi e le proprieta' che sono importanti per l'azienda, senza riguardo alla loro rappresentazione in un particolare sistema computerizzato utilizzando un software particolare, abbiamo quello che ci riguarda con, l'implementazione indipendente dall'analisi dei dati.
E' giusto chiedersi perche' l'analisi dei dati dovrebbe essere fatta se e' possibile, in pratica, andare direttamente ad una progettazione del sistema computerizzato. L'analisi dei dati e' perdita di tempo e ci pone un sacco di domande.
L'attuazione puo' essere rallentata mentre sono ricercate le risposte. E ' piu' opportuno avere un esperto analista "che vada con il lavoro" e terminare subito con il progetto. La principale differenza e' che l'analisi dei dati e' piu' probabile che porti a un progetto che soddisfi i requisiti presenti e futuri, che sia piu' facilmente adattabile ai cambiamenti nel business o nelle apparecchiature informatiche. Si puo' anche sostenere che tende a garantire che la politica dell'organizzazione dati deve avere risposta dai responsabili dell'organizzazione, non dagli analisti di sistemi. L'analisi dei dati puo' essere pensata come un approccio lento e attento e che omettendo questo passo diventa veloce e non precisa.
Da un altro punto di vista, l'analisi dei dati fornisce spunti utili per i principi generali di progettazione che andranno a beneficio dell'analista tirocinante anche se alla fine si arrivera' una soluzione rapida e non precisa.
Lo sviluppo di tecniche di analisi dei dati hanno contribuito a comprendere la struttura e il significato dei dati nelle organizzazioni.
La tecnica di analisi dei dati puo' essere utilizzata come primo passo per estrapolare la complessita' del mondo reale in un modello che puo' essere memorizzato su un computer ed essere accessibile da molti utenti.I dati possono essere raccolti con metodi convenzionali come intervistando persone dell'organizzazione e lo studio dei documenti. I fatti possono essere

rappresentati come oggetti di interesse.
Sono disponibili una serie di strumenti di documentazione per analizzare i dati, quali diagrammi entita' relazione. Questi aiuti sono utili per comunicare, contribuiscono a garantire che il lavoro si svolga in modo approfondito, e facilitano i processi di mappatura che seguono l'analisi dei dati. Alcune dei documenti possono essere utilizzati come documenti origine per il dizionario dei dati.
Nell'analisi dei dati si analizzano i dati e si costruisce una rappresentazione dei sistemi in forma di un modello di dati (concettuale). Un modello di dati concettuale indica la struttura dei dati e dei processi che utilizzano tali dati.

- Analisi dei dati = stabilisce la natura dei dati.
- Analisi funzionale = stabilisce l'utilizzo dei dati.

Tuttavia, poiche' i dati e le analisi funzionali sono cosi' mescolate, uso il termine analisi dei dati per coprire entrambe.
Costruire un modello di un'organizzazione non e' facile. L'intera organizzazione e' troppo grande in quanto ci sono troppe cose da modellare. Ci vuole troppo tempo ed e' facile non ottenere nulla di concreto come un sistema informativo, i manager vogliono risultati concreti abbastanza rapidamente. E' quindi compito dell'analista dei dati di modellare una particolare visione della organizzazione, quella che si rivela ragionevole e precisa per la maggior parte delle applicazioni e degli usi.
I dati hanno una loro intrinseca struttura, indipendente dal trattamento dal formato delle stampe ecc. Il modello dei dati mira a rendere esplicita questa struttura
L'analisi dei dati e' stata utilizzata per stabilire la natura e l'uso di dati.

Ciclo di vita dell'analisi dei dati



Quando un progettista di database si avvicina al problema di costruire un sistema database, i passaggi logici seguiti sono quelli del ciclo di vita dell'analisi del database:

Studio del database

Il progettista crea una specifica scritta per il sistema di database da costruire. Si tratta di:

- Analizzare la situazione dell'azienda - sia che sia una societa' in espansione, dinamica nei sui requisiti o matura nella natura o con solida esperienza nella formazione dei dipendenti per i nuovi prodotti interni, ecc Questi fattori hanno un impatto sul modo di essere visualizzati nella specifica.
- Definire i problemi ed i vincoli - qual e' la situazione attuale? Come reagisce l'azienda con chi deve eseguire i compiti connessi con il nuovo database. Eventuali problemi intorno al metodo attuale? Quali sono i limiti del nuovo sistema?
- Definire gli obiettivi - cosa e' il novo sistema di database che andiamo a fare, e in che modo deve essere fatto. Quali sono le informazioni che l'azienda desidera in concreto memorizzare, e che cosa desidera calcolare. Come si evolvono i dati.
- Definire gli scopi ed i confini - cio' che e' memorizzato su questo nuovo sistema di database e cosa e' conservato altrove. Si interfaccera' ad un altro database?

Progettazione del database

- E' una fase concettuale.
  Sono i passi di progettazione logica e fisica nell'acquisire le specifiche fisiche dei progetti implementabili.
  Questo si vedra' piu' da vicino tra un momento.

Implementazione e caricamento

- Caricamento e collaudo del software DBMS
  E' possibile che il database sia eseguito su una macchina che ancora non abbia al momento un sistema di gestione database in esecuzione. Se questo e' il caso, uno deve essere installato su quella macchina. Una volta che un DBMS e' stato installato, il database deve essere creato all'interno del DBMS.
  Infine, non tutti i database iniziano completamente vuoti, e quindi deve

essere caricato con l'iniziale set di dati (come ad esempio l'inventario corrente, nomi dell'attuale staff, I dettagli di alcuni clienti, ecc.)

Verifica e valutazione

- Il database, una volta implementato, deve essere testato contro le specifiche fornite dall'utente.
  E' utile anche per testare il database utilizzare dati fittizi, dati di cui gli utenti non sempre hanno una piena comprensione di che cosa hanno indicato e quello ottenuto di quanto differisce da quello che hanno effettivamente chiesto!
  Inoltre, questo passaggio nel ciclo di vita offre l'opportunita' al progettista di mettere a punto il sistema per le migliori prestazioni.
  Infine, e' un buon sistema per valutare il database in-situ, insieme a tutte le applicazioni collegate.

Operativita'

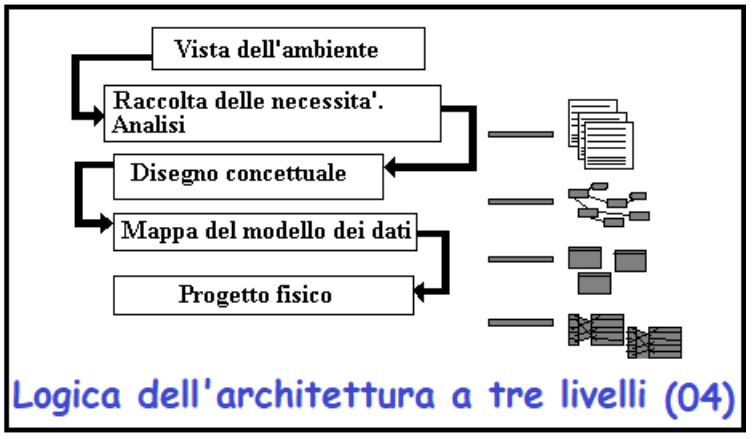
- Il database viene reso disponibile all'utenza.
  Questo passaggio e' quello in cui il sistema e' effettivamente in uso reale da parte della societa'.
  Comprende anche l'addestramento dell'utenza e la definitiva stesura dei manuali operativi

Manutenzione ed evoluzione

- Viene anche fatta una attenta analisi dei pareri degli utenti
  I progettisti raramente ottengo inizialmente cose perfette, e puo' esserci il caso che la societa' chieda dei cambiamenti per risolvere dei problemi con il sistema o per raccomandare miglioramenti o nuove esigenze.
  Comunemente lo sviluppo avviene senza modificare la struttura del database. nei sistemi piu' anziani la struttura DB si fossilizza.

Modello del database a tre livelli

Spesso indicato come il modello a tre livelli, questo e' dove il progettista si muove da una specifica scritta presa dai requisiti del mondo reale ad un progetto fisicamente implementabile per uno specifico DBMS. I tre livelli sono comunemente indicati come 'Progetto concettuale', ' Modello di mappatura dei dati', e 'Progetto fisico'.



db04.png

La specifica e' di solito sotto forma di un documento scritto contenente le esigenze dei clienti, i rapporti finiti, il progetto delle videate e simili, scritto dal cliente per indicare i requisiti che deve avere il sistema finale.
Spesso, tali dati devono essere raccolti e raggruppati da una varieta' di fonti interne alla societa' e poi analizzati per verificare se i requisiti sono necessari, corretti ed efficienti.
Una volta che i requisiti di database sono stati raccolti, la fase di progettazione concettuale prende le richieste e produce un modello di alto livello della struttura dei dati del database.
In questo modulo, si usa la modellazione ER per rappresentare i modelli di alto livello dei dati, ma ci sono altre tecniche. Questo modello e' indipendente dal DBMS finale su cui il database verra' installato.
Successivamente la fase di progettazione concettuale acquisisca il modello di alto livello dei dati e lo trasforma in uno schema concettuale, che e' specifico per una particolare classe di DBMS (es. relazionale).
Per un sistema relazionale, come Oracle, un appropriato schema concettuale sarebbero le relazioni.
Infine, nella fase di progettazione fisica lo schema concettuale viene convertito in strutture interne del database. Questo e' specifico per ogni particolare prodotto DBMS.

Modello del database Entità-Relazione (E-R)

E' un modello concettuale, e come tale fornisce una serie di costrutti (strutture), che descrivono la realtà in una maniera intuitiva che prescinde dai criteri di organizzazione fisica dei dati.
Il modello E-R usa simboli grafici per favorire la comprensione.
Gli schemi E-R sono essenzialmente dei grafici con aggiunte di frasi di specifica

e di vincolo.
Ogni entità rappresenta una classe di oggetti sia materiali che immateriali del mondo reale è caratterizzata da un nome e viene rappresentata con un rettangolo.

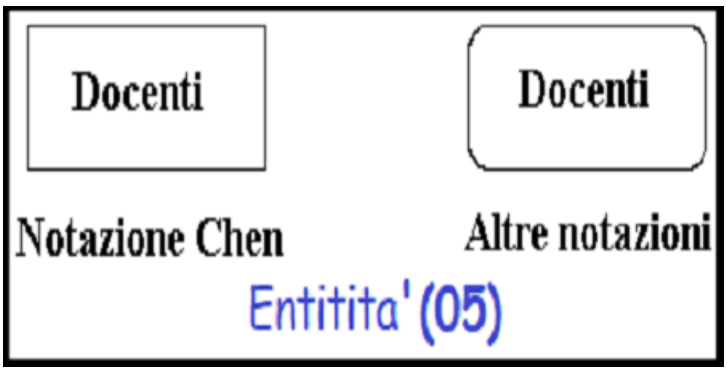
Nozioni di base

Modello delle relazioni tra entita' (Entity Relationship - ER)

- E' uno strumento di progettazione
- E' una rappresentazione grafica del sistema di database
- Fornisce un modello concettuale dei dati di alto livello.
- Supporta la percezione dei dati da parte degli utenti
- E' indipendente dall'hardware e dal DBMS
- Ha molte varianti
- E' composto da entita', attributi e relazioni

Entita'

- Una entita' sono le informazioni su un qualsiasi oggetto nel sistema che vogliamo modellare e memorizzare
- I singoli oggetti sono chiamati entita'
- Gruppi degli oggetti dello stesso tipo sono chiamati tipi di entita' o insiemi di entita'
- Le entita' sono rappresentate da rettangoli (sia con angoli rotondi o quadrati) **Figura 05**
- Ci sono due tipi di entita'; tipi di entita' deboli e forti.



ey05.png

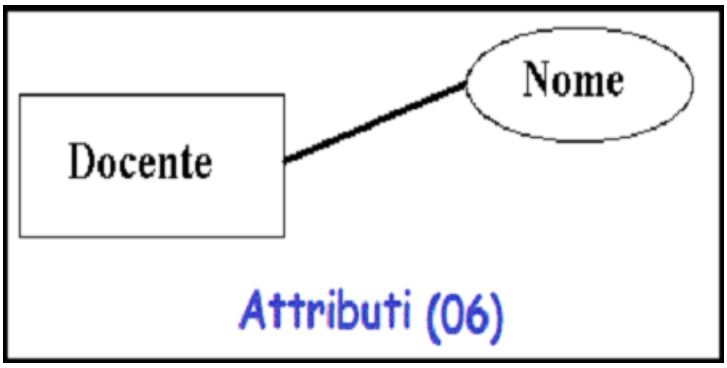
Attributi

- Tutti i dati relativi ad una entita' sono tenuti nei suoi attributi.

- Un attributo e' una proprieta' di un'entita'.
- Ciascun attributo puo' avere qualsiasi valore dal suo dominio.
- Ogni entita' all'interno di un tipo di entita':
    - Puo' avere un numero qualsiasi di attributi.
    - Puo' avere valori di attributo diversi che in qualsiasi altra entita'.
    - Hanno lo stesso numero di attributi.

  Gli attributi possono essere

    - Semplice o composto
    - A valore singolo o multivalore
- Gli attributi possono essere visualizzati su modelli E-R
- Essi appaiono all'interno di ovali e sono attaccati alla loro entita'.
- Si noti che i tipi di entita' possono avere un gran numero di attributi
  Se tutti sono disegnati gli schemi sarebbero confusi. Mostrano solo un attributo se si aggiunge informazioni al diagramma ER, o chiarisce il punto.**Figura 06**



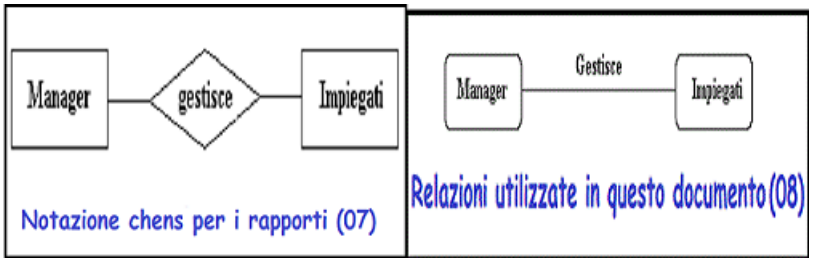
ey06.png

Chiavi

- Una chiave e' un dato che ci permette di identificare in modo univoco singole occorrenze o un tipo di entita'.
- Una chiave candidata e' un attributo o un insieme di attributi che identificano in modo univoco singole occorrenze o un tipo di entita'.
- Un tipo di entita' puo' avere uno o piu' possibili chiavi candidate, quella che e' selezionata e' conosciuta come la chiave primaria.
- Una chiave composita e' una chiave candidata che consiste di due o piu' attributi.
- Il nome di ogni attributo di chiave primaria e' sottolineato.

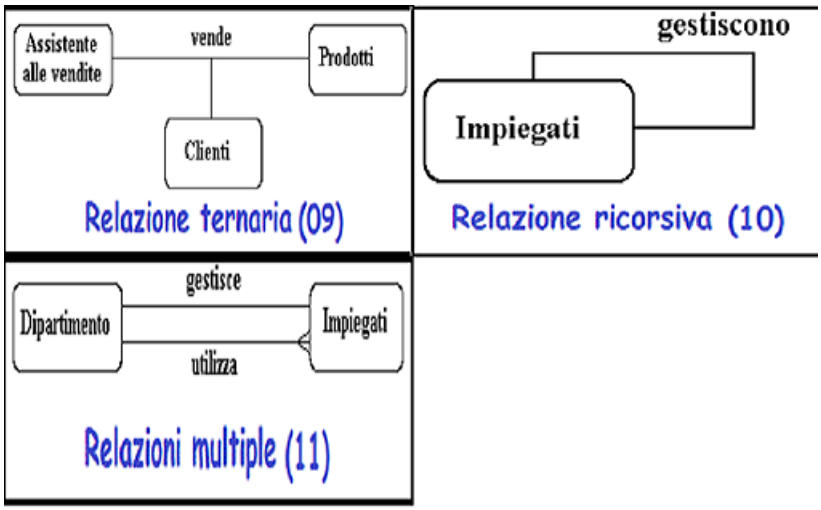
Relazioni

- Un tipo di relazione e' una associazione significativa tra i tipi di entita'
- Una relazione e' un'associazione di soggetti in cui l'associazione include una sola entita' da ogni tipo di entita' partecipante.
- I tipi di relazione sono rappresentate nel diagramma ER da una serie di linee.
- Come sempre, ci sono molte notazioni oggi in uso
  Nella notazione originale Chen, il rapporto e' inserito all'interno di un diamante, esempio i manager gestiscono i dipendenti: **Figura 07**
- Per questo modulo, uso una notazione alternativa, dove il rapporto e' un'etichetta sulla linea. Il significato e' identico. **Figura 08**



Grado di una relazione

- Il numero dei soggetti che partecipano in una relazione e' noto come il grado delle relazioni.
- Se ci sono due tipi di entita' coinvolte si tratta di una tipo di rapporto binario **Figura 08**
- Se ci sono tre tipi di entita' coinvolte si tratta di un tipo di relazione ternaria **Figura 09**
- E' possibile avere una relazione n-aria (ad esempio quaternaria o unitaria).
- Rapporti unitari sono anche noti come rapporti ricorsivi.**Figura 10**
- E' un rapporto in cui la stessa entita' partecipa piu' di una volta in ruoli diversi.
- Nel precedente esempio dico che i dipendenti sono gestiti dai dipendenti.
- Se volessimo maggiori informazioni su chi gestisce chi, si potrebbe introdurre un secondo tipo di entita' chiamata manager.
- E 'anche possibile avere entita' associate attraverso due o piu' relazioni distinte. **Figura 11**
- Nella rappresentazione che usiamo non e' possibile avere gli attributi come parte di un rapporto. Devono essere sviluppati diversi altri tipi entita' a supporto di cio'.

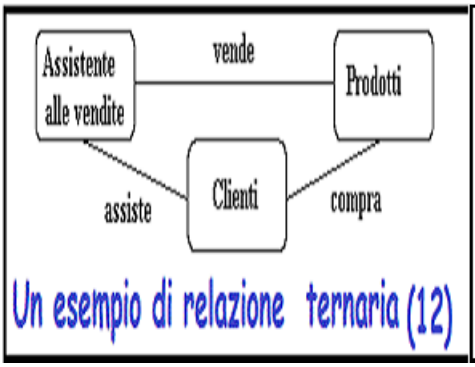


ey09-10-11.png

I rapporti ternari - valutazione e rimozione.

Quando le relazioni ternarie si verificano in un modello ER devono sempre essere rimosse prima di terminare il modello. A volte le relazioni possono essere sostituite da una serie di relazioni binarie che collegano le coppie delle relazioni del ternario originale. **Figura 12**

- Cio' puo' provocare la perdita di alcune informazioni - Non e' piu' chiaro quale assistente alle vendite ha venduto a un cliente o un particolare prodotto.
- Provare a sostituire il rapporto ternario con un tipo di entita' e una serie di relazioni binarie.
  Le relazioni sono di solito i verbi, per cui il nome del nuovo tipo di entita' dal verbo di relazione viene riscritto come un sostantivo.
- La relazione *vendere,* puo' diventare la relazione tipo *vendita*.**Figura 13**
- Cosi' un assistente alle vendite puo' essere collegato a uno specifico cliente e tutti e due alla vendita di un prodotto particolare.
- Questo processo funziona anche per le relazioni di ordine superiore.

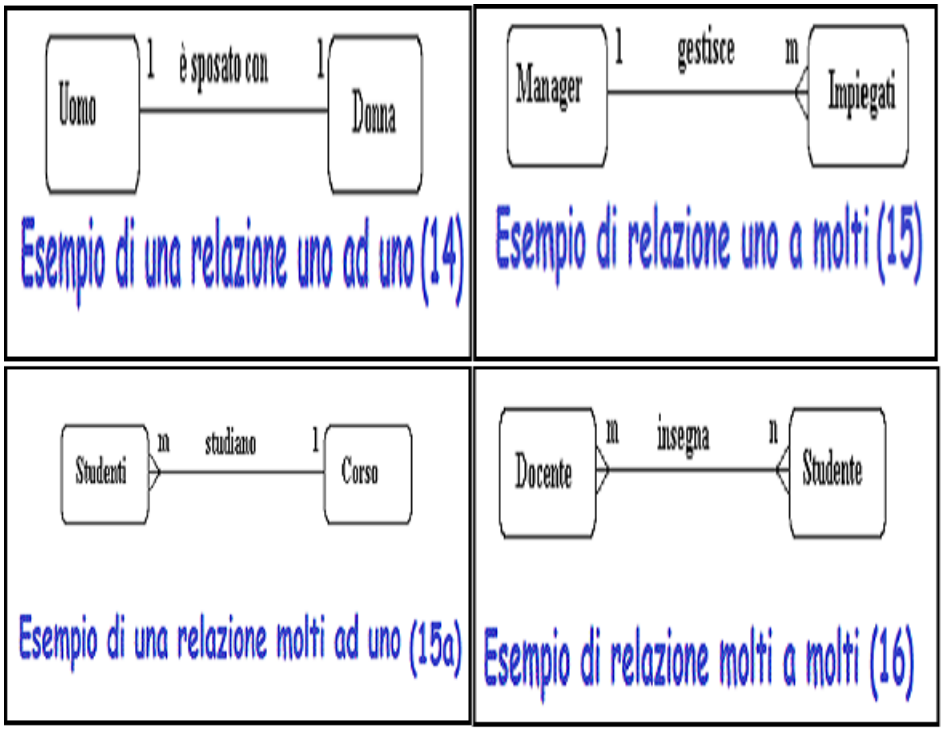


Un esempio di relazione ternaria (12)

Sostituzione di un rapporto ternario (13)

Cardinalita'

- Le relazioni sono raramente uno ad uno
  Per esempio, un manager gestisce di solito piu' di un dipendente
- Questo e' descritto dalla relazione di cardinalita', per queste sono possibili quattro categorie.
  - Relazione uno a uno (1:1)
    Un uomo puo' sposare solo una donna, ed una donna puo' sposare solo un uomo, quindi e' una relazione 1-1 (1:1).**Figura 14**
  - Relazione uno a molti (1: m)
    Un manager gestisce molti dipendenti, ma ogni dipendente ha un solo manager, quindi si tratta di una relazione uno a molti (1: m)**Figura 15**
  - Relazione molti a uno (m: 1)
    Molti studenti per un corso di studio. Non studiano piu' di un corso, quindi e' una relazione molti a uno (m: 1)**Figura 15a**
  - Relazione molti a molti (m: m)
    Un docente insegna a molti studenti e uno studente impara da docenti diversi, quindi e' una relazione molti a molti (m: m). **Figura 16**
- Su un diagramma ER, se la fine di una relazione e' dritta, rappresenta uno, mentre una fine "a piede di corvo" rappresenta molti.



ey14.png

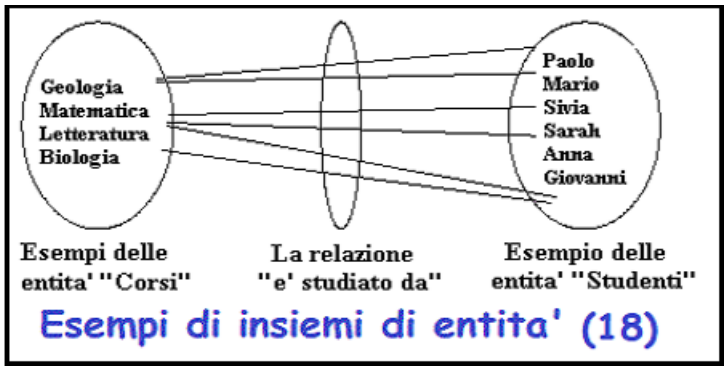
Opzionalita'

Una relazione puo' essere facoltativa o obbligatoria.

- Se la relazione e' obbligatoria
  Un'entita' ad una estremita' della relazione deve essere correlata a un'entita' al all'altra estremita'.
- L'opzionalita' puo' essere diverso a ciascuna estremita' della relazione
  Per esempio, uno studente deve essere inserito in un corso. Questo e' obbligatorio. Per la relazione "studente studia in un corso" e' obbligatoria.
- Ma un corso puo' esistere prima che gli studenti si siano iscritti. Cosi' il rapporto "il corso e' studiato da uno studente" e' opzionale.
- Per mostrare l'opzionalita', mettere un cerchio o un "O" alla "fine opzionale" della relazione.
- Dato che la relazione "il corso e' studiato dagli studenti " e' opzionale, e la parte facoltativa di cio' e' lo studente, allora la "O" va alla fine della connessione della relazione studente.**Figura 17**
- E' importante conoscere le opzionalita', perche' e' necessario assicurarsi che ogni volta che si crea una nuova entita' abbia i necessari collegamenti obbligatori.

Insiemi di entita'

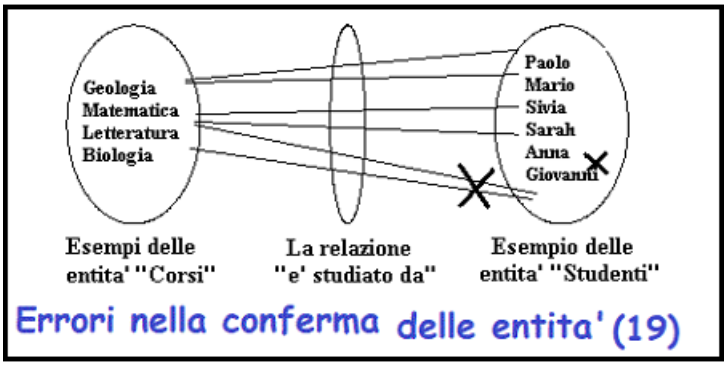
A volte e' utile provare vari esempi di entita' in un modello ER. Una ragione per questo e' per confermare la corretta cardinalita' e l'opzionalita' di una relazione. Io uso un "diagramma di insieme di entita'" per mostrare graficamente esempi di entita'. Si consideri l'esempio di "il corso e' studiato dallo studente".**Figura 18**



La conferma della correttezza

Figura 19

- Utilizzare il diagramma per illustrare tutti gli scenari di una possibile relazione.
- Tornare alla specifica dei requisiti e controllare per vedere se sono permessi.
- In caso contrario, mettere una croce sulle relazioni proibite
- Questo permette di mostrare la cardinalita' e l'opzionalita' delle relazioni.



db19.png

Derivare i parametri della relazione

Per controllare che i parametri siano corretti (a volte conosciuto anche come il grado) di una relazione, facciamo due domande:

1. Un corso e' studiato da quanti studenti? Risposta = "zero o piu".
   - Questo ci da' la laurea alla fine degli "studenti".
   - La risposta `zero o piu' 'deve essere diviso in due parti.
   - La parte "more" significa che la cardinalita' e' "molti".
   - La parte "zero" significa che il rapporto e' "optional".
   - Se la risposta e' "uno o piu", allora il rapporto sarebbe " obbligatorio".
2. Uno studente quanti corsi studia? Risposta = "UNO"
   - Questo ci da' la laurea alla fine del "corso" della relazione.
   - Questa risposta "UNO" significa che la cardinalita' di questo rapporto e' 1, ed e' "obbligatoria"
   - Se la risposta e' stata "zero o uno", allora la cardinalita' della relazione sarebbe stata 1, e sarebbe "optional"

Relazioni ridondanti

Alcuni diagrammi ER finiscono con una relazione ciclica (loop).

- Controllare per vedere se e' possibile spezzare il ciclo senza perdere informazioni
- Dati tre soggetti A, B, C, dove ci sono i rapporti AB, BC e CA, controllare se e' possibile navigare tra A e C via B. Se e' possibile, allora A-C e' una relazione rapporto.

- Controllare sempre con attenzione il modo per semplificare il diagramma ER.
  Rende piu' facile leggere le informazioni rimanenti.

Esempi di relazioni ridondanti

Considerare l'entita' "cliente" (dati di un cliente), "indirizzo" (l'indirizzo di un cliente) e "distanza" (distanza tra l'azienda e l'indirizzo del cliente).**Figura 20**

Dividere le relazioni m:m

Una relazione molti a molti in un modello ER non e' necessariamente sbagliata. Possono essere sostituite usando una entita' intermedia.
Questo dovrebbe essere fatto solo se:

- La relazione m:m nasconde un'entita'
- il diagramma ER risultante e' piu' facile da capire.

Esempio
Si consideri il caso di una societa' di autonoleggio. Un cliente noleggia molte auto e una macchina viene noleggiata da molti clienti.**Figura 21**
La relazione molti a molti puo' essere suddivisa per rivelare una "entita'" noleggio, che contiene un attributo "data di noleggio".**Figura 22**

Relazioni ridondanti (20)

Esempio di molti a molti (21)

Esempio della divisione molti a molti (22)

Costruire un modello ER

Prima di iniziare a disegnare il modello ER, leggere con attenzione la specifica dei requisiti. Documentare tutte le eventuali ipotesi che e' necessario fare.

1. Identificare le entita'
    - Elencare tutti i tipi di entita' potenziali. Questi sono l'oggetto dell'interesse nel sistema. E' meglio mettere tutte le possibili entita' in questa fase e poi, se necessario, scartarle in seguito.
2. Rimuovere le entita' duplicate
    - Assicurarsi che i tipi di entita' siano veramente separati o dare due nomi alla stessa cosa.
3. Non comprendere il sistema come un tipo di entita'
    - Esempio se la modellazione riguarda una libreria, i tipi di entita' possono essere i libri, i manuali, ecc La biblioteca e' il sistema, quindi non dovrebbe essere una entita'
4. Elencare gli attributi di ogni entita' (tutte le proprieta' per descrivere le entita' che sono rilevanti per l'applicazione).
    - Garantire che i tipi di entita' siano realmente necessari.
    - E' uno di loro attribuito solo di un altro tipo di entita'?
    - Se e' cosi' tenerli come attributi e incrociarle fuori dalla lista entita'.
    - Non hanno gli attributi di un soggetto come attributi di un'altra entita'
5. Segnare le chiavi primarie.
    - Cosa permette si identificare in modo univoco le istanze di quel tipo di entita'?
    - Questo potrebbe non essere possibile per alcuni soggetti deboli.
6. Definire le relazioni
    - Esaminare ogni tipo di entita' per vedere il suo rapporto con le altre.
7. Descrivere la cardinalita' e opzionalita' delle relazioni
    - Esaminare i vincoli tra le entita' partecipanti.
8. Rimuovere le relazioni ridondanti
    - Esaminare il modello ER per i rapporti ridondanti.

La modellazione ER e' un processo iterativo, in modo da disegnare diverse versioni, raffinando ognuna finche' non si e' contenti. Si noti che non esiste una giusta risposta al problema, ma alcune soluzioni sono meglio di altre!

Estratto da "https://www.electroyou.it/mediawiki/index.php?title=UsersPages:Carlopavana:dbms-i-database-relazionali"