Archivi e database

Archivi

Un archivio è un insieme organizzato di informazioni caratterizzate da:

- nesso logico tra informazioni;

- formato, ovvero la maniera in cui i dati sono interpretati e rappresentati (ne rende possibile l'interpretazione es. 123, abc);

- sono registrate su un supporto in cui è possibile scrivere e rileggere informazioni nel tempo;

- organizzazione, ovvero il modo con cui i record vengono sistemati nel supporto che li contiene (sono organizzate in modo da permettere una facile consultazione).

Es. elenco telefonico

Nell'archivio le informazioni vengono raggruppate secondo unità logiche.

Record: insieme di queste informazioni logicamente organizzate;

Campi: singole informazioni che compongono il Record.

Tracciato record: elenco dei Campi.

Archivi: insieme di record.

Memorie

Il disco magnetico è raggruppato in cluster.

Le memorie di massa sono caratterizzate da:

-Tipo di accesso: diretto, random o sequenziale.

-Capacità: quantità di dati che il supporto può contenere.

-Access time: intervallo di tempo (ms) tra la richiesta di trasferimento di dati e l'inizio del trasferimento verso la memoria centrale.

-Transfert rate: velocità di trasferimento dei dati dalla memoria di massa a quella centrale, è il numero di byte trasferiti nella memoria del computer in un secondo (KB/s).

Organizzazione degli archivi

Organizzare i dati è necessario per:

-consentire il loro ritrovamento in tempi brevi, per ottimizzare l'uso dello spazio disponibile sul supporto e per ridurre i tempi di accesso e di ritrovamento dei dati.

Esistono 3 tipi di organizzazione: sequenziale, relative e ad indici.

Organizzazione sequenziale:

consiste nel registrare i record uno di seguito all'altro, in lettura essi possono essere ritrovati scorrendo tutti i record del file partendo dal primo all'ultimo.

PRO: consente l'uso di record con lunghezza diversa.

CONTRO: diventa poco efficiente con molti record.

Caratteri speciali: EOR (end of record, separa i record), EOF (end of file, fine archivio).

Organizzazione relative (o diretta):

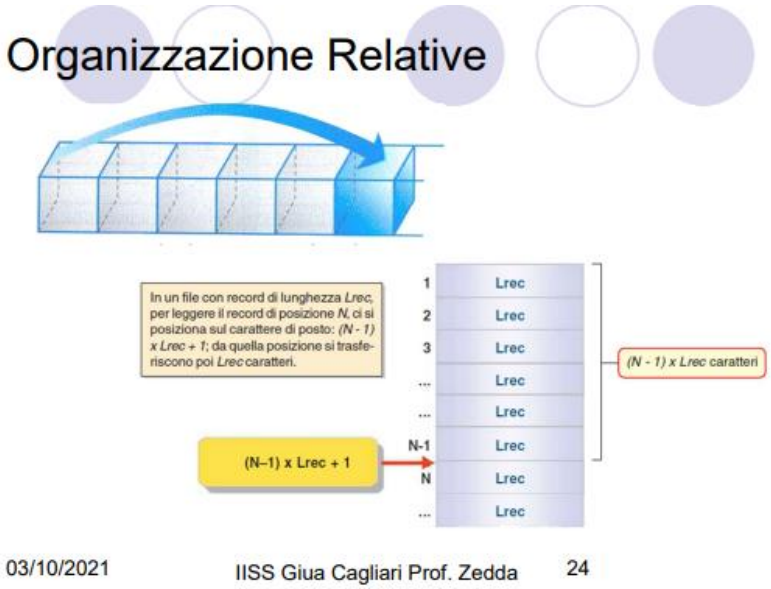
consiste nell'associare ad ogni record un numero d'ordine per trovare nel file il numero del primo carattere del record richiesto (se vengono inseriti uno di seguito all'altro il n dei record viene assegnato aggiungendo una unità). È utile quando i record vengono identificati attraverso un numero d'ordine progressivo. Se non può essere reperito con un numero si utilizza una chiave.

PRO: ci permette di accedere in maniera diretta al record che ci interessa e non occorre leggere tutti i record.

CONTRO: bisogna sapere la posizione di esso.

LR=lunghezza record; NR=numero record

Carattere: (NR-1)*LR+1.



Organizzazione ad indici (indexed)

La usiamo quando un record all'interno di un archivio non può essere reperito attraverso un numero. Utilizziamo infatti, una chiave ovvero una variabile alfanumerica univoca per ogni record che non ammette duplicati (codice fiscale etc..).

Un file quindi, è composto dall'area delle chiavi (indice) e dall'area dei dati.

Ad ogni chiave è associato il puntatore al record, cioè la posizione che il record occupa nel file. Si accede al record solo specificando la chiave.

Per accedere al record, le chiavi devono essere ordinate in modo sequenziale e si cerca la chiave sempre in modo sequenziale (magari con la ricerca binaria).

Si può accede al record in maniera sequenziale o relative.

Inoltre, si può scorrere il file partendo non necessariamente dal primo record.

L’identificazione di un record viene fatta automaticamente dal sistema sulla base della chiave del record (nell’area indice del file sul disco si trovano le informazioni che consentono di puntare al record desiderato).

Ad ogni record possiamo associare più chiavi (primaria o alternativa).

Le operazioni possono essere effettuate solo usando la chiave primaria, e sono:

-Inserimento

-Aggiornamento

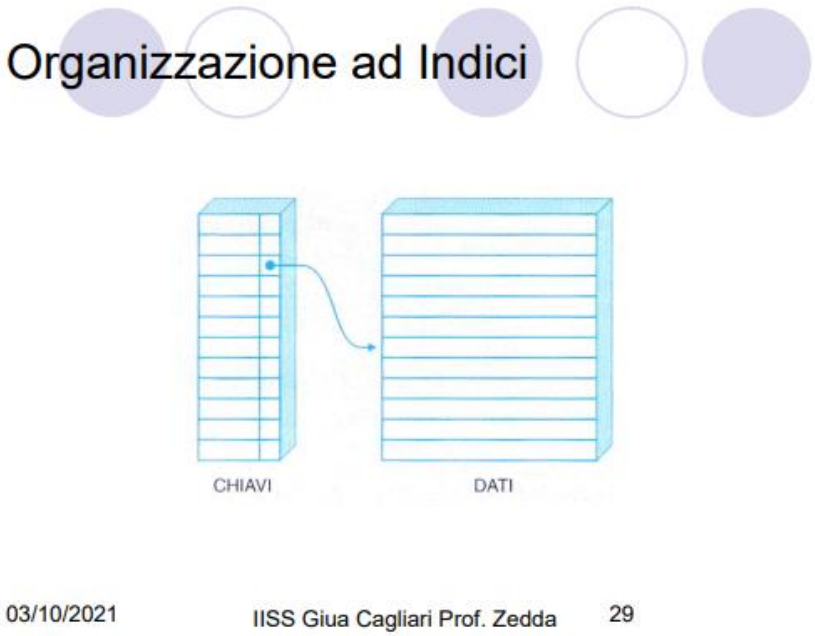
-Cancellazione.

Gli accessi possono essere:

-*Sequenziale*: operazioni I/O fatte in base all’ordine crescente delle chiavi, adatta per la ricerca sequenziale;

-*Random*: si può puntare direttamente al record desiderato quando si ha il valore della chiave;

-*Dinamico*: permette di alternare accessi sequenziali a altri random (molto flessibile), ad esempio è possibile posizionarsi su un record corrispondente alla chiave fornita e proseguire con l’accesso ai record successivi in modo sequenziale.



Differenza tra organizzazione e accesso degli archivi:

L'organizzazione fa riferimento a come i record vengono fisicamente strutturati nella memoria, mentre l'accesso indica come sarà possibile reperire il dato record

Limiti delle applicazioni basate su archivi:

-Ridondanza, ovvero la risorsa potrebbe duplicarsi provocando complicazioni.

-Incongruenza, ovvero quando 2 duplicati sono uno diverso dall'altro.

-Inconsistenza, ovvero quando non rappresenta la realtà.

-Rigidità dell'accesso al dato

-Dipendenza stretta tra applicazione e archivio, una modifica dell'archivio comporta una modifica nell'applicazione.

-Isolamento dei dati

-Difficoltà nell'esprimere vincoli di integrità

-Difficoltà nel gestire la concorrenza, cioè quando si cerca di far leggere/scrivere le informazioni da più applicazioni o utenti.

-Riservatezza del dato non granulare.

Database

Un database è una collezione di archivi di dati ben organizzati e ben strutturati, in modo che possano costituire una base di lavoro per utenti diversi con programmi diversi. È organizzato in modo integrato attraverso tecniche di modellazione dei dati e gestiti sulle memorie di massa attraverso appositi software, con l'obbiettivo di raggiungere una grande efficienza nel trattamento e nel ritrovamento dei dati, superando i limiti presenti nelle organizzazioni degli archivi.

I Database hanno molti vantaggi rispetto agli archivi, come:

-Indipendenza dalla struttura fisica dei dati: si possono modificare i supporti dei dati e le modalità di accesso senza modificare le applicazioni;

-Utilizzo da parte di uno o più utenti: il DBMS permette che le operazioni svolte da diversi utenti non interferiscano l'una con l'altra;

-Eliminazione della ridondanza: gli stessi dati compaiono più volte in archivi diversi;

-Eliminazione dell'inconsistenza: il database non può avere campi uguali con valori diversi in archivi diversi;

-Facilità di accesso: i dati sono più facili e veloci da trovare;

-Integrità dei dati: vengono attuati dei controlli per evitare anomalie ai dati;

-Sicurezza dei dati: vengono impediti accessi non autorizzati ai dati grazie a delle procedure di controllo;

-Uso di linguaggi per la gestione del database: il database viene gestito attraverso comandi di manipolazione dei dati per ottenere informazioni desiderate.

Gestione del DataBase

Il DBMS organizza un database, ed è un'interfaccia tra gli utenti di un database con le loro applicazioni e le risorse costituite dall'hardware e dagli archivi di dati presenti in un sistema di elaborazione.

Funzioni del DBMS:

1-**Implementazione del modello logico sul sistema di elaborazione**, definendo i dati e le strutture di dati derivate dallo schema del modello logico, le viste ottenute attraverso proiezioni e congiunzioni e organizzando fisicamente i dati sui supporti.

2-**Manipolazione e interrogazione sulla base dei dati**, ovvero le operazioni di inserimento, modifica o cancellazione; l'interfaccia tra programmi degli utenti e l'accesso dei dati del db attraverso interfacce.

3-**Controllo dell'integrità dei dati**. Esse sono: integrità sulle entità, integrità referenziale e integrità definite dall'utente.

4-**Sicurezza e protezione**, ovvero la protezione dei dati dai malfunzionamenti hw e sw con la possibilità di recovery in caso di perdita dei dati; autorizzazione degli utenti che accedono al db e protezione dagli accessi non autorizzati.

Linguaggi DB

Comandi:

-DDL (data definition language): crea i dati nelle tabelle e ordina al DBMS di creare una struttura fisica nel DB;

-DCL (data control language): stabilisce le autorizzazioni (tipi di permessi) e i vincoli;

-DML (data manipulation language): manipola i dati per la modifica/cancellazione/o inserimento dei record;

-QL (query language): si occupa di interrogare il database, consentendone il ritrovamento dei dati.

Utenti

-Amministratore del DB (DBA, (DDL, DCL)): gestisce il database e implementa il DB nel sistema di elaborazione sui supporti fisici delle memorie di massa; gestisce i dati, le autorizzazioni degli accessi; definisce delle viste parziali di utenti al DB; controlla la disponibilità degli spazi sulla memoria di massa e anche gli interventi di recupero in caso di cattivi funzionamenti.

-Programmatori: utilizzano di DML e agisce solamente sotto il controllo del gruppo di progetto e di produzione del SW.

-Utenti Finali: accedono al DB attraverso i comandi di interrogazione o attraverso interfacce SW.

Transazione: è l'insieme delle operazioni di interrogazione o modifica che devono essere eseguite come se fossero un'unica operazione.

ACID

Le proprietà che un DBMS dovrebbe sempre garantire per ogni transazione, fanno parte dell'ACID (atomicità, consistenza, isolamento, durabilità).

-Atomicità: la transazione è un'entità atomica indivisibile e quindi la sua esecuzione non può essere parziale, ma totale o nulla.

-Consistenza: garantisce che al termine dell'esecuzione di una transazione, i vincoli di integrità sono soddisfatti.

-Isolamento: gli effetti delle transazioni devono essere indipendenti l'uno dall'altro.

-Durabilità: gli effetti di una transazione devono essere memorizzati permanentemente nel database.

2) Livello logico

La modellazione dei dati è una procedura che si suddivide in più parti.

1)Realtà, analisi di un sistema (insieme di elementi (sottoinsiemi) di natura diversa che interagiscono per raggiungere determinati obbiettivi) e astrazione. Andiamo a trovare gli elementi fondamentali della nostra classe;

2)Concettuale (semplificazione, creazione diagramma delle classi con DIA);

3)Logico (tracciato record)

4) Fisico

Modello relazionale

È chiamato così perché fondato sul concetto matematico di relazione tra insiemi di oggetti. Si basa su alcuni concetti fondamentali e ha l'obbiettivo di utilizzare il linguaggio matematico (linguaggio conosciuto da tutti) e di eliminare i problemi di ambiguità nella terminologia e nella simbologia.

Quando andiamo a creare degli insiemi, ognuno di loro sarà un **dominio,** ovvero l'insieme dei valori che il campo d'istanza può assumere (n dominio=n colonne, ognuna definisce un attributo). Dal numero di domini deriverà il grado della **relazione** (es. 2 domini, relazione di secondo grado, 2 colonne), che è un sottoinsieme del prodotto cartesiano tra n insiemi (semplificazione). Per trovare tutte le combinazioni possibili, eseguiamo il **prodotto cartesiano** tra gli insiemi (es. 3*5*6). Ogni riga è chiamata N-uple, e l'insieme delle righe è chiamata **cardinalità**.

Colonne: domini (grado); righe: N-uple (cardinalità).

La relazione rappresenta una classe, ogni n-upla rappresenta un'istanza della classe, le colonne (attributi) contengono i campi di istanza della classe (nome della colonna), il domino è l'insieme dei valori che possono essere assunti da un attributo.

Chiave: è un attributo o una combinazione di attributi che identificano univocamente le n-uple.

Il modello relazionale di un DB è un insieme di tabelle (relazioni) sulle quali si possono effettuare operazioni, e tra le quali possono essere stabilite associazioni.

a. Tutte le righe della tabella contengono lo **stesso numero di colonne**, corrispondenti agli attributi.

b. Gli attributi rappresentano **informazioni elementari** (o atomiche), non scomponibili ulteriormente, cioè non ci sono campi di gruppo che contengono per ogni riga un insieme di valori anziché un solo valore.

c. I valori assunti da un campo appartengono al dominio dei valori possibili per quel campo, e quindi sono **valori omogenei** tra loro, cioè sono dello stesso tipo.

d. In una relazione, ogni riga è diversa da tutte le altre, cioè non ci possono essere due righe con gli stessi valori dei campi: questo significa che esiste un attributo o una combinazione di più attributi che identificano univocamente la n-upla, e che assumono perciò la funzione di **chiave primaria** della relazione.

e. Le n-uple compaiono nella tabella secondo un **ordine non prefissato**, cioè non è rilevante il criterio con il quale le righe sono sistemate nella tabella.

Uno a uno: diventa un'unica relazione che contiene i campi della prima e della seconda classe.

Uno a molti: l'id della classe di partenza diventa l'FK della relazione di arrivo associata.

Molti a molti: si crea una relazione di collegamento composta dagli id univoci delle due entità e dagli attributi dell'associazione. Bisogna creare una relazione unica che contiene sia gli attributi della super-classe che quelli delle sottoclassi.

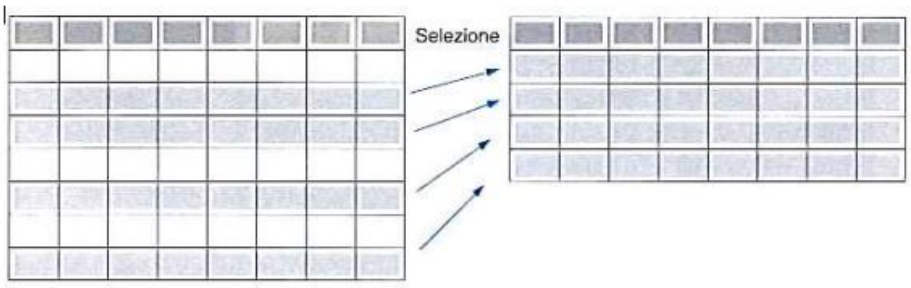
Operatori relazionali

Agiscono su una relazione per crearne una nuova: consentono di effettuare le interrogazioni al DB per ottenere informazioni desiderate (o estraendo una sotto-tabella da una tabella, o combinando tra loro due o più tabelle).

Operatori: select, project, join.

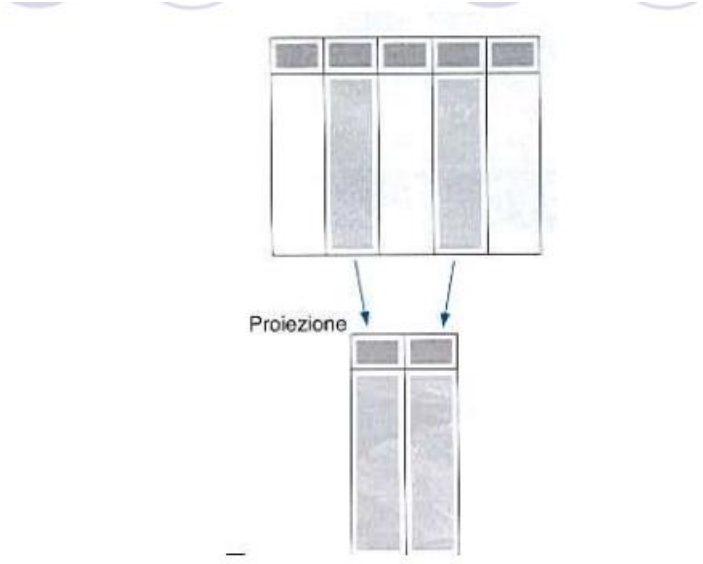
Select (selezione): nuova relazione con stesso numero di colonne ma diverso numero di righe.

Quindi stesso grado ma diversa cardinalità (N-uple <=).



Project (proiezione): nuova relazione con stesso numero di righe ma diverso numero di colonne.

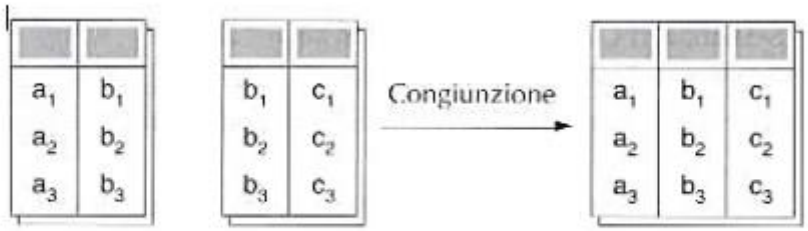
Quindi stessa cardinalità (N-uple) ma diverso grado.



Join (congiunzione): nuova relazione che viene realizzata dalla combinazione delle 2 relazioni aventi uno o più attributi in comune. Contiene le righe della prima e della seconda tabella.

I gradi N1 e N2 diventano nella nuova relazione N1+N2-1, perché l'attributo comune compare una sola volta, mentre la cardinalità non è prevedibile.

La congiunzione viene chiamata **equi-join** perché viene realizzata facendo corrispondere valori uguali per attributi comuni nelle due tabelle



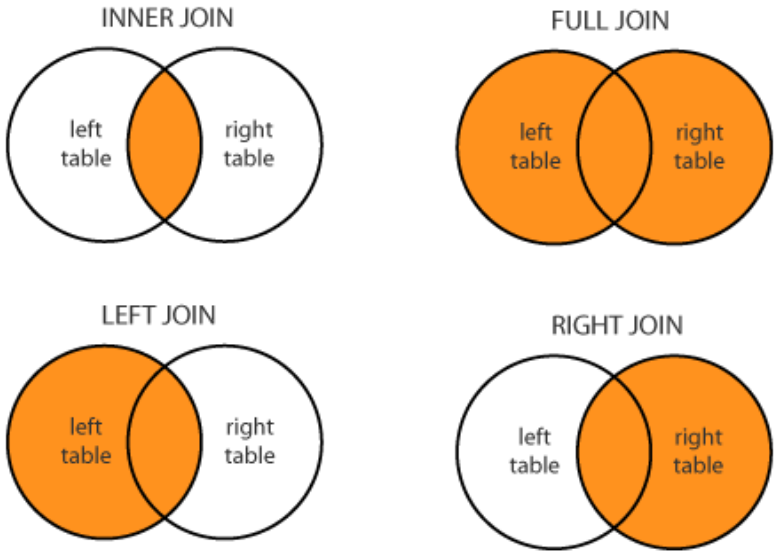
Join esterno: restituisce le righe di entrambe le tabelle e vengono combinate solo quando il valore dell'attributo comune nella prima tabella trova un valore uguale nella colonna dell'attributo comune della seconda tabella.

Tipi di join esterno:

-Left join: elenca tutte le righe della prima tabella congiungendo tra le righe della seconda solo quelle per le quali si trovano valori corrispondenti per l'attributo comune.

-Right join: restituisce tutte le righe della seconda tabella e congiunge con le righe della prima tabella solo quelle per le quali si trovano valori corrispondenti.

-Full join: restituisce tutte le righe.



Self join: si attua quando c'è una gerarchia nella stessa classe. Vengono combinate righe di una tabella con le righe della stessa tabella quando sono presenti valori corrispondenti per attributi, ovvero due attributi con lo stesso dominio.

Es. Abbiamo 4 operai ognuno con un proprio id e Marco, id 1, è il capo cantiere. Verrà quindi creata una FK nella stessa tabella OperaioId e sarà tutto impostato a 1 perché ogni operaio ha come capo cantiere Marco.

IdOp	**Nome**	**OpId**
1	*Marco*	*1*
2	*Diego*	*1*
3	*Pippo*	*1*
4	*Pluto*	*1*

Normalizzazione

È una procedura con la quale le tabelle vengono trasformate in modo tale che ognuna corrisponda a un singolo oggetto della realtà (database). Le sue regole sono definite per evitare l'inconsistenza e anomalie.

Essa è utile per la creazione di tabelle ben definite che facilitano le operazioni di aggiunta, modifica e cancellazione delle informazioni e rendono possibili eventuali cambiamenti nella struttura del modello. La normalizzazione deve evitare la perdita di dati a seguito della scomposizione di una tabella e deve evitare l'inconsistenza.

La definizione di relazioni normalizzate avviene a livelli crescenti di normalizzazione (prima, seconda e terza).

Chiave (PK): è l'insieme di uno o più attributi che identificano in modo univoco una N-upla.

Chiave candidata: è l'insieme di uno o più attributi che possono svolgere la funzione di chiave (ce ne possono essere più di una).

Chiave alternativa: è una qualsiasi chiave candidata che non è stata scelta come chiave primaria.

Attributo non-chiave: è un campo che non fa parte della chiave primaria.

Integrità referenziale: verifica l'integrità tra PK e FK.

Dipendenza funzionale e transitiva

Funzionale: si ha quando il valore di un attributo A1 determina un singolo valore dell'attributo A2.

Quindi: A2 dipende funzionalmente da A1; A1 è un determinante per A2.

A1→A2

Transitiva: si ha quando un attributo A2 dipende da A1 e l'attributo A3 dipende da A2; allora A3 dipende transitivamente da A1.

A1→A2 A2→A3 A1→A3

Prima forma normale (1FN)

Una relazione si dice tale quando rispetta i requisiti fondamentali del modello relazionale:

-tutte le righe contengono lo stesso numero di colonne;

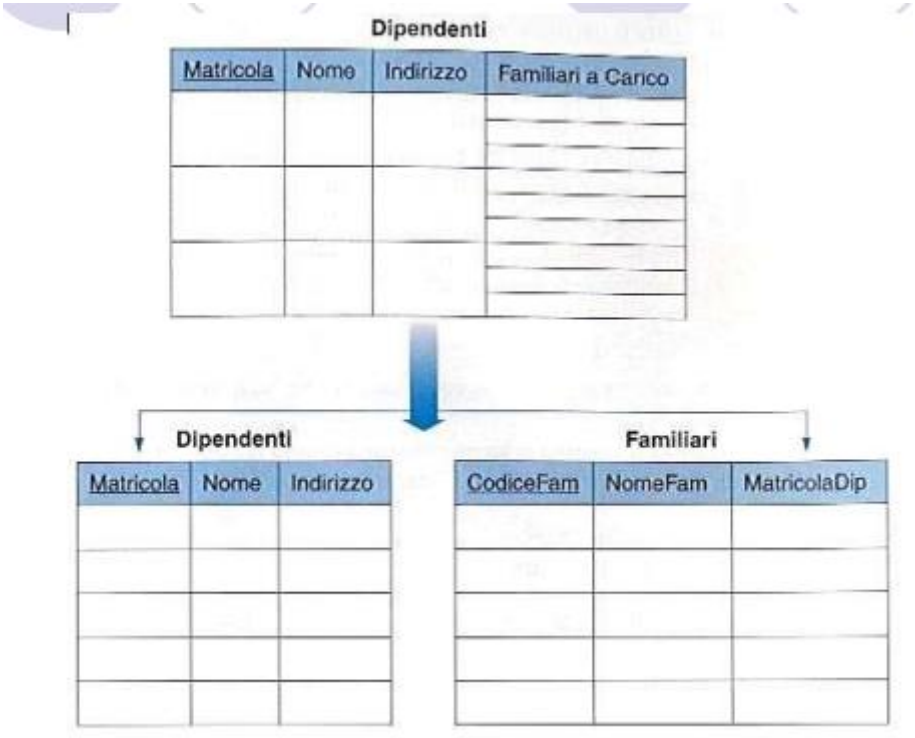
-ogni attributo è elementare (gli attributi non devono essere ulteriormente scomponibili);

-i valori che compaiono in una colonna sono dello stesso tipo (appartengono allo stesso dominio);

-ogni riga è diversa dalle altre (non ci possono essere due righe con gli stessi valori nelle colonne);

-l'ordine con il quale le righe compaiono nella tabella è irrilevante.

La 1FN può provocare problemi durante le operazioni di aggiornamento o cancellazione e quindi provocare inconsistenza o perdita di dati.



Le forme superiori alla prima vengono introdotte per eliminare problemi durante le operazioni di aggiornamento o cancellazione.

Seconda forma normale (2FN)

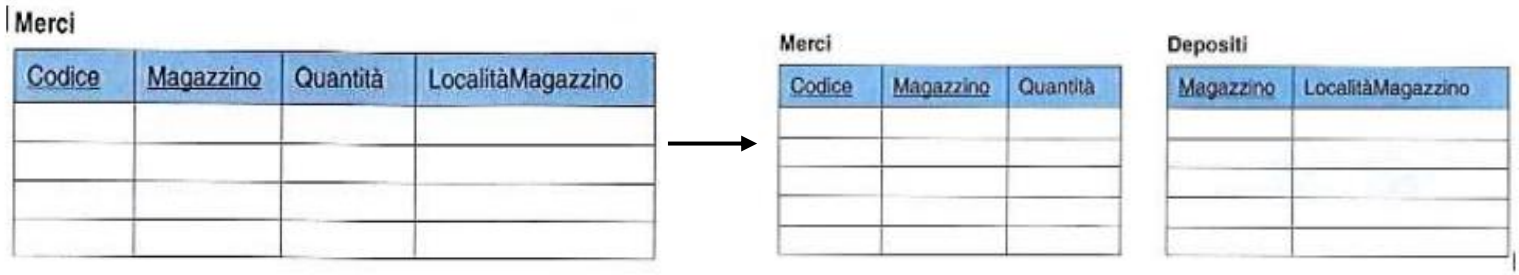
Una relazione è tale quando è in 1FN e non ci sono attributi non-chiave che dipendono parzialmente dalla chiave. In un caso con chiave primaria composta, gli attributi non chiave dipendono solo da una parte di essa.

Es. Merci (Codice, Magazzino, Quantità, Località Magazzino)

Chiave composta, rischia di provocare problemi.

Soluzione:

Merci (Codice, Magazzino, Quantità) / Depositi (Magazzino, Località Magazzino).

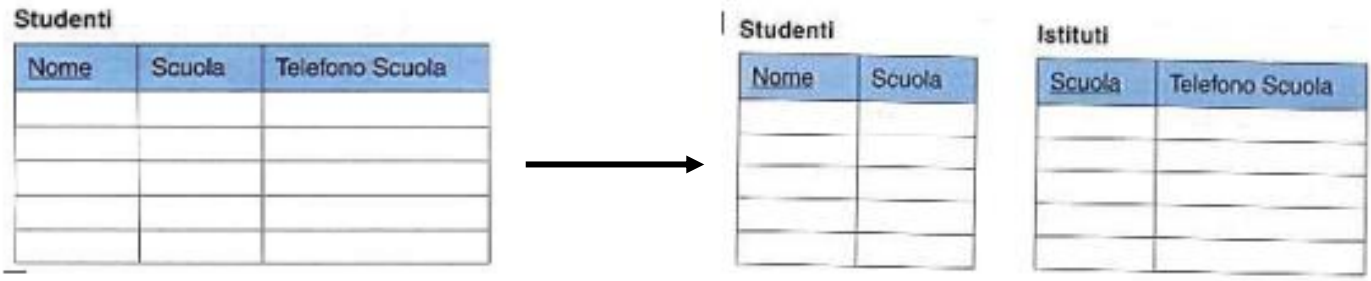


Problemi: possibilità di ridondanza e inconsistenza.

Terza forma normale (3FN)

Una relazione si dice tale quando è in 2FN e tutti gli attributi non-chiave dipendono direttamente dalla chiave, cioè non possiede attributi non-chiave che dipendono da altri attributi non-chiave. La terza forma normale elimina la dipendenza transitiva dagli attributi dalla chiave.

La 3FN si ottiene scomponendo la relazione di partenza in due nuove relazioni, nelle quali tutti gli attributi dipendono direttamente dalla chiave, togliendo gli attributi non-chiave che dipendono da un altro attributo non chiave.



Riassunto forme normali

1FN: una relazione si dice tale quando rispetta i requisiti fondamentali del modello relazionale, in particolare ogni attributo è elementare, non ci sono righe uguali e non ci sono attributi ripetitivi.

2FN: una relazione è tale quando è in 1FN e non ci sono attributi non-chiave che dipendono parzialmente dalla chiave.

3FN: è tale quando è in 2FN e non ci sono attributi non-chiave che dipendono transitivamente dalla chiave.

BCFN: è tale quando è in 2FN e in essa ogni determinante è una chiave candidata.

Integrità Referenziale

È un insieme di regole del modello relazionale che garantiscono l'integrità dei dati quando si hanno relazioni associate tra loro attraverso la chiave esterna: queste regole servono per rendere valide le associazioni tra le tabelle e per eliminare gli errori di inserimento, cancellazione o modifica di dati collegati tra loro.

Viene rispettata quando per ogni valore NON NULLO della chiave esterna, esiste un valore corrispondente della chiave primaria nella tabella associata.

Regole di eliminazione

Eliminazione a cascata (o a catena): si attua quando la cancellazione di un elemento di una classe influenza pure l'elemento della classe collegata.

Esempio: cancelliamo uno studente, verranno eliminate anche le verifiche che ha sostenuto.

Eliminazione restrittiva: si può eseguire solamente rimuovendo prima tutto ciò collegato a essa.

Esempio: si potrà cancellare uno studente solo dopo aver eliminato tutte le verifiche da lui sostenute.

Tecniche di indicizzazione

Multilivello: l'organizzazione a indici su più livelli cerca di risolvere problemi di inefficienza nell'organizzazione del file riducendo la dimensione dello spazio di ricerca per gli indici. Il file degli indici viene a sua volta indicizzato costruendo un indice (indice di livello 2) per accedere al file degli indici vero e proprio (indice di livello 1), in una posizione vicina a quella dell'indice cercato. Se anche l'indice di livello 2 è troppo grande si può costruire un indice di livello 3 per accedere all'indice di livello 2 che punta all'indice vero e proprio.

Albero binario: partendo da una radice, inseriamo il numero seguente a destra se è maggiore e a sinistra se è minore. Partendo da questo, si continua, ritrovandosi una struttura efficiente e bilanciata

Metodo Hash: metodo bisogna prendere la chiave che voglio indicizzare e trasformo i caratteri in ascii, avremo un codice numerico che va sommato. Decidiamo poi quante n-uple mantenere nel nostro DB (ad esempio 20) e prendiamo il numero primo più vicino (23), dividiamo il numero sommato per il numero primo e otterremo un resto tra 0 e 22, quello sarà il valore associato al campo. Scegliere un numero alto per evitare collisioni. Le n-uple indicizzate con hash che collidono, vengono poste in un'altra area, nell'area di overflow.

Blazor:

Blazor è un framework free e opensource che permette di sviluppare applicazioni web completamente in C# senza ricorrere a Javascript ma all'occorrenza garantisce l'interoperabilità con JS. In questo modo sia front-end che back-end, sono sviluppate nello stesso linguaggio e questo dà la possibilità di condividere le stesse condividere le stesse classi.

Esistono 2 tipi di Blazor: **WebAssembly e Blazor Server**. Per sviluppare webapp con accesso a database utilizziamo Blazor Server, per non complicare l'architettura.

Il linguaggio permette di scrivere codice C# combinato con codice html è **razor**.

Ado.Net: è un insieme di classi che permettono alle applicazioni .NET di interagire con diverse basi di dati. Esistono diversi Provider di dati specializzati a lavorare con una specifica base di dati. Noi usiamo SQLite col provider Microsoft.Data.Sqlite accessibile tramite NuGet. Ma esistono provider per Access, SQLServer, Oracle, MySql, MariaDB e tanti altri.

Le classi Ado.Net sono:

Connection: apre e chiude la connessione col database e contiene il percorso del db (@Data Source= Data/chinook.db);

Command: manda in esecuzione una query con i metodi Executereader (restituisce un datareader), Executescalar (restituisce un dato semplice), Executenonquery (restituisce il numero delle righe delle tabelle coinvolte.

Parameter: crea un parametro. Se la query ha dei parametri, per ognuno di essi viene definita un'istanza di Parameter;

Datareader: ospita in RAM i risultati della query. È in sola lettura e legge una riga per volta col metodo read che restituisce true se il datareader non è finito.

Un component lo vediamo come un tag personalizzato, che fa ciò che vogliamo noi. Gestisce sia parte del codice che parte grafica. AtPage: quel component può essere raggiunto con un percorso.

Tutto ciò su blazor è un component. Sono composti dall'interfaccia e codice c#.

Componente Routable: innesta tutti i componenti. Può essere ricercato nella route, esempio @page pippo: cercando nella route possiamo accedere al component.

Componente non routable: non può essere ricercato nella route, ma può essere utilizzato per l'elaborazione di altri componenti, esempio <table>.