

I MANGA DELLE SCIENZE

DATABASE

MANA TAKAHASHI
SHOKO AZUMA
TREND-PRO CO., LTD.





I MANGA DELLE SCIENZE

DATABASE

MANA TAKAHASHI
SHOKO AZUMA
TREND-PRO CO., LTD.



SOMMARIO

PREFAZIONE	IX
1	
CHE COS'È UN DATABASE?	1
A che cosa servono i database?.....	2
Che succede nel regno?.....	16
I dati vengono duplicati.	16
I dati possono entrare in conflitto	17
I dati sono difficili da aggiornare	18
Un database: ecco la soluzione!	19
Come si usa un database	19
Riassumendo	21
2	
CHE COS'È UN DATABASE RELAZIONALE?	23
La terminologia	24
I database relazionali.....	34
I modelli di database	39
Operazioni di estrazione dei dati.....	39
Gli operatori di insieme	39
Gli operatori relazionali	43
Domande	45
Il database relazionale vince!	47
Riassumendo	48
Risposte	48
3	
PROGETTIAMO UN DATABASE!	49
Il modello E-R	50
Normalizzare una tabella.....	56
Che cos'è il modello E-R?	74
Come si analizza il modello E-R	74
Caso 1: associazioni uno-a-uno	74
Caso 2: associazioni uno-a-molti	75
Caso 3: associazioni molti-a-molti	75
Domande	76
Normalizzare una tabella.....	78
Domande	79
I passi per progettare un database	81
Riassumendo	81
Risposte	82

4

IMPARIAMO IL LINGUAGGIO SQL!	85
SQL: come usarlo	86
La ricerca dei dati	93
L'estrazione dei dati.	98
Unire le tabelle.	101
Creare una tabella	103
Panoramica su SQL	106
La ricerca dei dati con l'istruzione SELECT.	106
Stabilire le condizioni	107
Gli operatori di confronto	107
Operatori logici	107
I pattern	108
Le ricerche.	108
Domande	109
Le funzioni di aggregazione.	110
Aggregare i dati con l'operatore di raggruppamento	110
Domande	111
La ricerca dei dati	112
Usare una subquery	112
Usare una subquery correlata.	113
Domande	114
Unire le tabelle.	114
Creare una tabella	115
Inserire, aggiornare o cancellare le righe	116
Creare una vista	117
Domande	118
Riassumendo	119
Risposte	119

5

FACCIAMO FUNZIONARE UN DATABASE!	125
Che cos'è una transazione?	126
Che cos'è un blocco?.	131
Sicurezza e database	138
L'indicizzazione	143
Ripristino in caso di malfunzionamento.	148
Le proprietà delle transazioni	153
Atomicità	153
Coerenza	154
Isolamento.	155
Durabilità.	159
In caso di disastro	161
Le tipologie di fallimento	161
I checkpoint.	161
Domande	162

Gli indici	162
Domande	164
Ottimizzare una Query	164
I cicli annidati	165
Sort Merge (ordina e unisci)	166
Hash	166
L'ottimizzatore	167
Riassumendo	167
Risposte	167
6	
I DATABASE SONO OVUNQUE!	169
I database in uso	175
I database e il web	177
Database distribuiti	183
Le procedure archiviate e i trigger	185
I database nel web	194
Usare le procedure archiviate	196
Domande	196
Che cos'è un database distribuito?	197
La distribuzione orizzontale	197
La distribuzione verticale	198
Partizionamento dei dati	198
Il partizionamento orizzontale	198
Il partizionamento verticale	199
Prevenire le incoerenze con un commit a due fasi	199
Domande	201
La replicazione di un database	201
La modalità di sola lettura	201
La replicazione con accesso a tutti i server	202
Ulteriori applicazioni dei database	202
XML	202
I database orientati agli oggetti	203
Riassumendo	205
Risposte	205
Conclusioni	205
APPENDICE	
LE ISTRUZIONI SQL PIÙ USATE	207
INDICE	209

PREFAZIONE

I database rappresentano una delle componenti fondamentali di tutti i sistemi aziendali informatici. Alcuni lettori di questo libro potrebbero prendere in considerazione l'idea di introdurre i database nella loro esperienza quotidiana di lavoro. Altri potrebbero avere l'esigenza di sviluppare sistemi basati su database. Il database è la tecnologia che supporta da dietro le quinte questi sistemi e non è semplice comprendere la sua vera natura.

Questo libro è concepito per consentirvi di imparare i rudimenti dei database attraverso la lettura di un manga. Alla fine di ogni capitolo troverete esercizi pratici di verifica e approfondimento delle nozioni che avrete appreso. Ogni capitolo è strutturato in modo da associare alla comprensione della tecnologia dei database la verifica pratica della vostra comprensione.

La struttura di questo libro è la seguente.

Il Capitolo 1 spiega perché usiamo i database. Perché i database sono necessari? Quali problemi dobbiamo affrontare se non usiamo i database? Apprenderete quindi informazioni di base sul loro utilizzo.

Il Capitolo 2 affronta la terminologia specifica. Scoprirete diversi modelli di database e altri termini legati ai database.

Il Capitolo 3 spiega come progettare un database, nello specifico un database relazionale (quello più diffuso).

Il Capitolo 4 è dedicato a SQL, un linguaggio usato per gestire i database relazionali. Grazie a SQL imparerete a gestire con facilità i vostri dati.

Il Capitolo 5 spiega la struttura di un sistema basato su database. Visto che consente di condividere dati tra molti utenti, imparerete a farlo nel modo più sicuro.

Il Capitolo 6 è dedicato alle applicazioni per database.

Imparerete come vengono usati i database Web-based o di altro tipo.

Questo libro è il frutto della collaborazione di molte persone: Shoko Azuma per il fumetto, TREND-PRO per la produzione e Ohmsha per la supervisione, l'editing e il marketing. A tutte le figure coinvolte rivolgo i miei ringraziamenti.

Spero che questo libro si rivelerà utile a tutti i lettori.

MANA TAKAHASHI

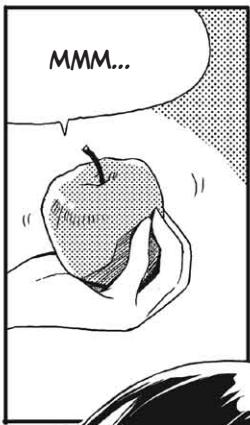
1

CHE COS'È UN DATABASE?











"IN UNA TERRA
LONTANA CHE ABBIAMO
VISITATO ABBIAMO
TROVATO UN LIBRO CHE
PARLA DI UNO STRUMENTO
RIVOLUZIONARIO.

LA PERSONA
CHE CE L'HA CONSEGNATO
CI HA DETTO CHE QUESTA
TECNICA SEGRETA SI
CHIAMA DATABASE.

ABBIAMO SENTITO
CHE IL DATABASE È UN
SISTEMA CHE PERMETTE
DI CONDIVIDERE, GESTIRE
E USARE I DATI.

MA COME VIENE
UTILIZZATO DIPENDE DA CHI
LEGGE QUESTO LIBRO.

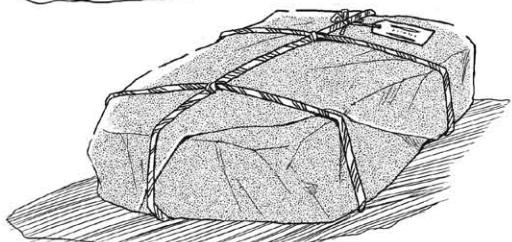
POI CI HA
CONSEGNAZIO IL LIBRO,
NELLA CONVINZIONE
CHE IL REGNO DI KOD
L'AVREBBE USATO PER
SCOPI PACIFICI.

RURUNA...

...USA QUESTO
LIBRO PER IL
PROGRESSO
DELLA NOSTRA
NAZIONE."

MA PER PIACERE!
VOI NON AVETE IDEA
DI QUANTO MI SENTO
STRESSATA!

STRAPP
STRAPP!!



MA CHE CAVOLO?!

OH, COM'È
VECCIO...

FRUSC

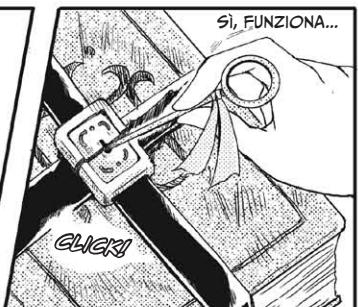
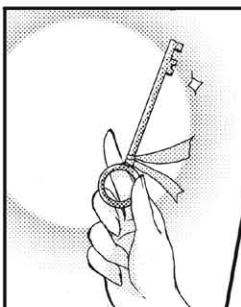
MA È CHIUSO.

NON SI APRE.

MM...

"FORSE CON
QUESTA CHIAVE?

L'HO TROVATA NELLA BUSTA...



SÌ, FUNZIONA...

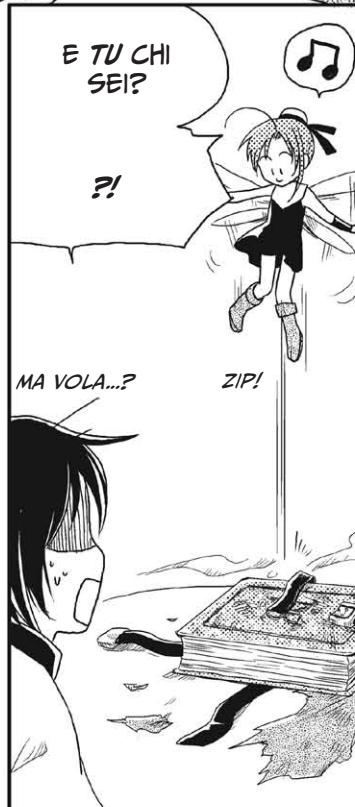
KA-BOOM!

AAAHHH!

?!
AGGI

COPP
COPP







BE', TUTTO
SOMMATO
SEMPRE
ABBASTANZA
INNOCUA...

GIÀ...

BASTA PARLARE
DI ME! AVETE
APERTO IL LIBRO
PER IMPARARE
TUTTO SUI
DATABASE...

...NON È COSÌ?

IMMAGINO
DI SÌ.

VA BENE,
ALLORA,
COMINCIA-
MO.

PER
CREARE UN
DATABASE...

!!!!

FERMA UN
MOMENTO!

SARÀ PURE
UNA DOMANDA
BANALE...

...MA CHE COS'È UN
DATABASE?

OH, NON
SAPETE
COSA SIA?

AVETE A CHE
FARE CON MOLTI
VALORI E MOLTI
NUMERI, GIUSTO?

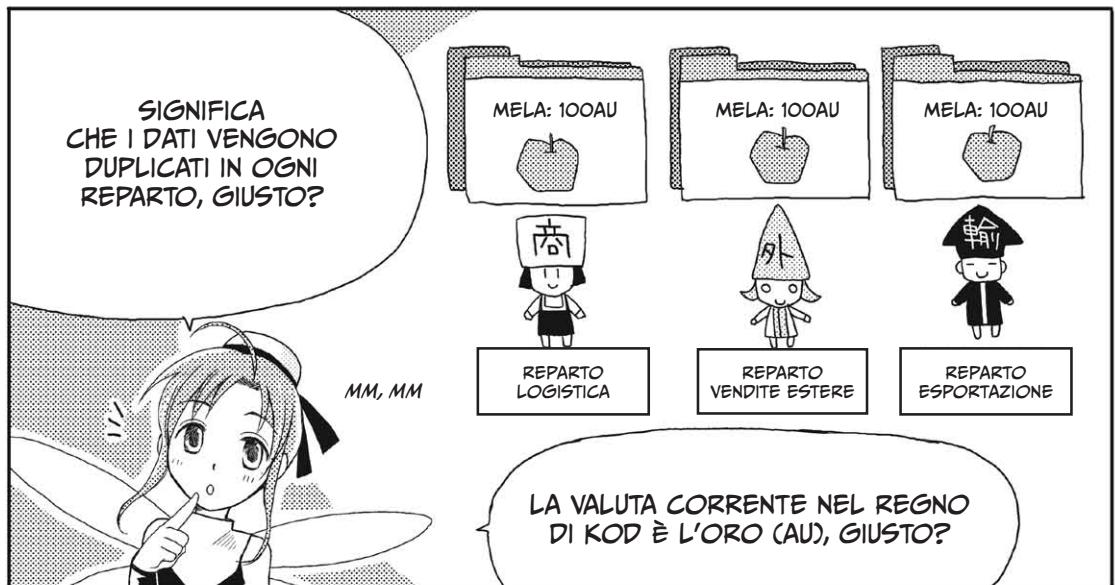
SÌ, E HO
MOLTI
PROBLEMI...

GESTISCO
NUMERI E VALORI
COLLEGATI
A DIVERSI
PRODOTTI,

CLIENTI E
VENDITE
GENERANDO FILE
A SECONDA DEI
REPARTI.

OH, QUINDI STAI
GESTENDO I DATI IN
MODO SCOORDINATO,
DIVISI PER REPARTO.

MMMH



LA VALUTA CORRENTE NEL REGNO
DI KOD È L'ORO (AU), GIUSTO?



GIÀ, QUANDO IL
PREZZO DELLE
MELE È SALITO.

GIÀ, È SALITO
DA 100AU A 120AU,
SE NON RICORDO
MALE.

IMPENNATA DEL PREZZO DI UNA MERCE

MM, MM

HO INFORMATO
TUTTI I
REPARTI DEL
Cambiamento di
PREZZO, MA...

AUMENTATE IL
PREZZO A 120AU!



MA...?

UNO DEI REPARTI
SI È SCORDATO DI
MODIFICARLO.

SHOCK!

NON HO
RICEVUTO
IL TUO
MESSAGGIO...

STAVO DORMENDO...

VENDITE
ESTERE

MELA 100AU

IL PREZZO RESTA COME PRIMA!



MIO PADRE HA DETTO,
"IN FUTURO DOVREMO
FAR VEDERE AI TURISTI LA
RACCOLTA DELLA FRUTTA!"

MA SECONDO
ME NON SIAMO
ANCORA PRONTI.

LA FA
SEMPLICE,
LUI.



ALLARGHIAMO GLI
AFFARI USANDO IL
NOSTRO SISTEMA
CONSOLIDATO.
AH AH AH!

SE DOVESSIMO
AMPLIARE GLI AFFARI
SAREBBE IMPOSSIBILE
GESTIRE I DATI CON IL
SISTEMA ATTUALE.

I DATI SI
MISCHIEREBBERO
TRA LORO...

MA... IL MIO
LAVORO NON
SI RIDURRA' DI
CERTO, COSÌ!

CHE RABBIA!

SI CALMI,
LA
PREGO!



SE APRISTE UN NUOVO
CANALE DI VENDITA
DOVRETE CREARE
NUOVI FILE PER IL
NUOVO REPARTO.

BE', ALLORA OGNI VOLTA
CHE CAMBIA QUALCOSA
DOVETE AGGIORNARE
E CONFIRMARE I DATI,
MI SEMBRA UNA BELLA
FATICA.

ANCHE SE FARETE DEL
VOSTRO MEGLIO, LA
GESTIONE DEI DATI
SARÀ SEMPRE UN
INCUBO, NON È COSÌ?

UN MUCCIO DI
DOCUMENTI





CHE SUCCIDE NEL REGNO?



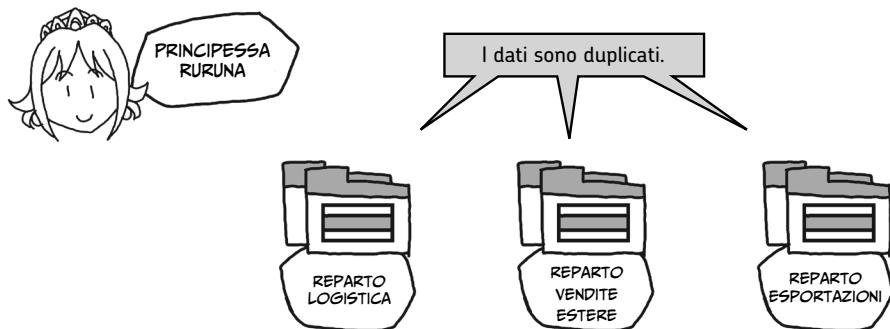
Il Regno di Kod gestisce i suoi dati con un sistema di file informatici. Ma sembra che questo sistema abbia qualche problema. Quali, nello specifico? Osserviamo il sistema da vicino.

Nel regno ci sono attualmente tre reparti: il Reparto Logistica, il Reparto Vendite Estere e il Reparto Esportazioni. Il Reparto Logistica tiene traccia di tutta la frutta prodotta nel Regno. Il Reparto Vendite Estere gestisce i contatti con i partner commerciali del Regno, mentre il Reparto Esportazioni registra tutte le spedizioni di frutta del Regno.

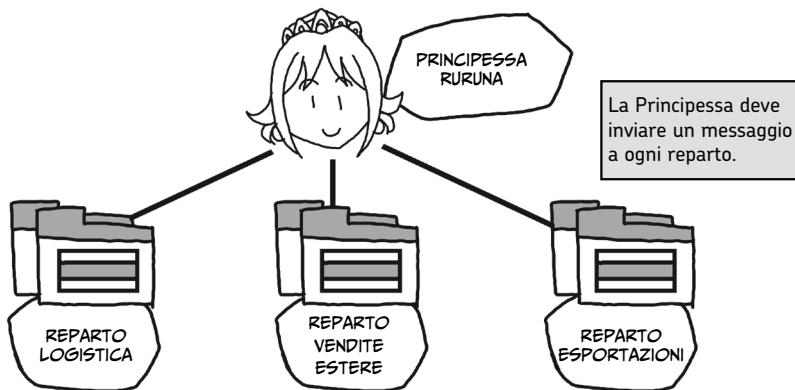


I DATI VENGONO DUPLICATI

Il sistema attuale non soddisfa la Principessa Ruruna. Perché? Ogni reparto del Regno gestisce i dati in maniera autonoma. Per esempio, il Reparto Logistica e il Reparto Esportazioni generano i propri file per gestire le informazioni sulla frutta. Questo significa che si crea un'inutile duplicazione dei dati tra i reparti. Ogni reparto deve inserire i dati, conservarli e stampare moduli di conferma, uno spreco insomma. In aggiunta, i dati posseduti da un reparto non sono mai condivisi con gli altri reparti.

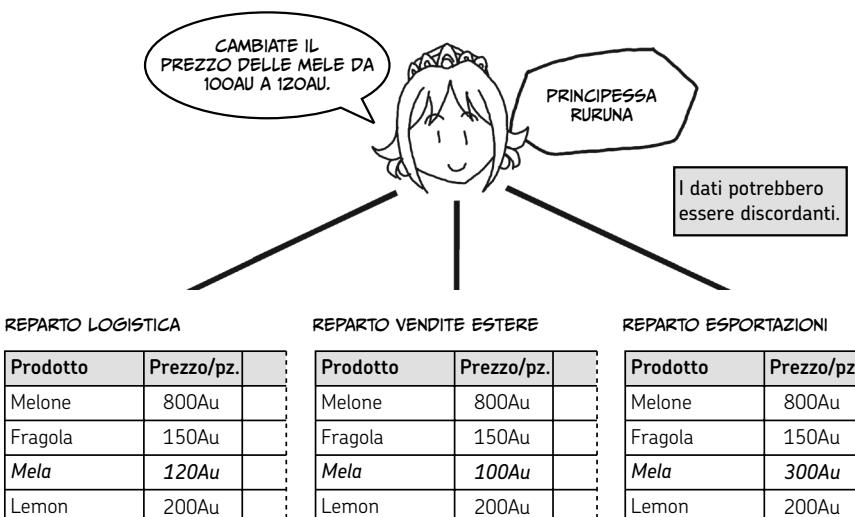


E non è tutto. Il sistema può generare problemi quando qualcuno deve modificare i dati. Ipotizziamo per esempio che il prezzo delle mele cambi. Per gestire questa modifica la Principessa Ruruna deve informare del cambiamento ogni reparto individualmente. Non è scomodo?



I DATI POSSONO ENTRARE IN CONFLITTO

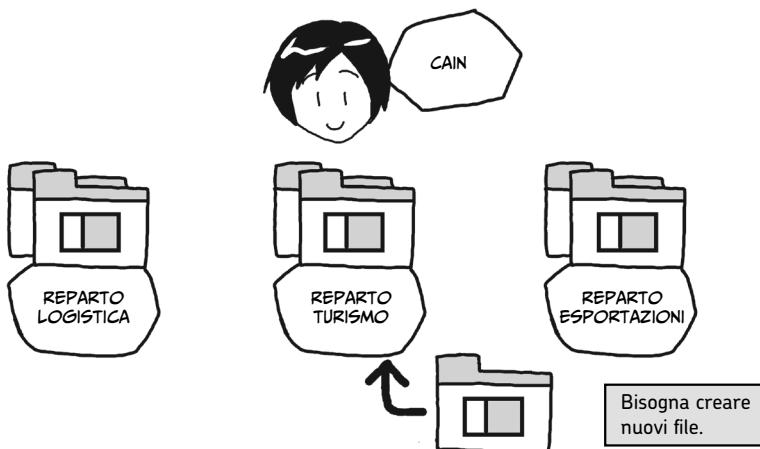
Potrebbe sembrare semplice notificare a tutti i reparti che il prezzo delle mele è cambiato, ma possono insorgere dei problemi. Ipotizziamo che la Principessa Ruruna comunichi ai tre reparti che il prezzo delle mele è cambiato. Il Reparto Vendite Estere, però, potrebbe dimenticarsi di modificare il prezzo, oppure il Reparto Esportazioni potrebbe portarlo a 300Au, invece che a 120Au. Questi errori provocano un conflitto tra i dati presenti nei reparti: il dato contenuto nel sistema è diverso dal dato reale. Un bel problema!



I DATI SONO DIFFICILI DA AGGIORNARE

Il sistema attuale non solo crea conflitti tra i dati, ma rende anche difficile rispondere alle esigenze del mercato. Per esempio, mettiamo che il Re voglia lanciare un nuovo Reparto Turismo. Quando una guida accompagnerà i turisti per i frutteti e li intratterrà con i dati di vendita del Regno, sicuramente vorrà avere accesso alle cifre più aggiornate disponibili.

Sfortunatamente il sistema attuale non consente a un reparto di accedere ai dati degli altri reparti, visto che i file sono gestiti individualmente. Per gestire il nuovo *business* turistico, la Principessa Ruruna dovrà trasferire tutti i dati rilevanti all'interno del Reparto Turistico!



FILE PER IL REPARTO LOGISTICA

Prodotto	Prezzo/pz.	
Melone	800Au	
Fragola	150Au	
Mela	120Au	
Limone	200Au	

FILE PER IL REPARTO TURISMO

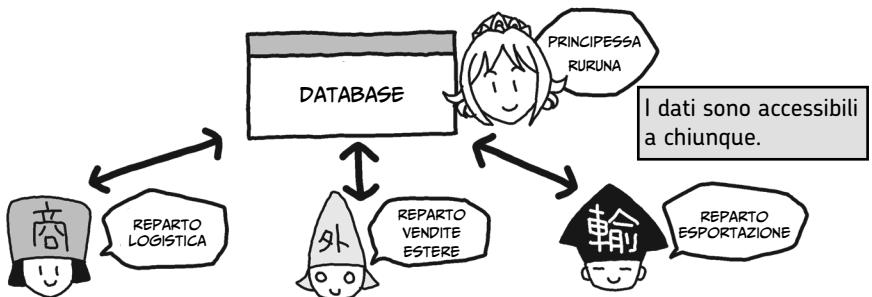
Prodotto	Prezzo/pz.	
Melone	800Au	
Fragola	150Au	
Mela	120Au	
Limone	200Au	

Questo, a sua volta, aumenta il rischio di duplicazioni ogni volta che nasce un nuovo reparto. Alla luce di questi punti deboli, il sistema attuale non è efficiente. Rende complessa l'apertura di nuovi progetti e la reattività agli stimoli esterni.

UN DATABASE: ECCO LA SOLUZIONE!



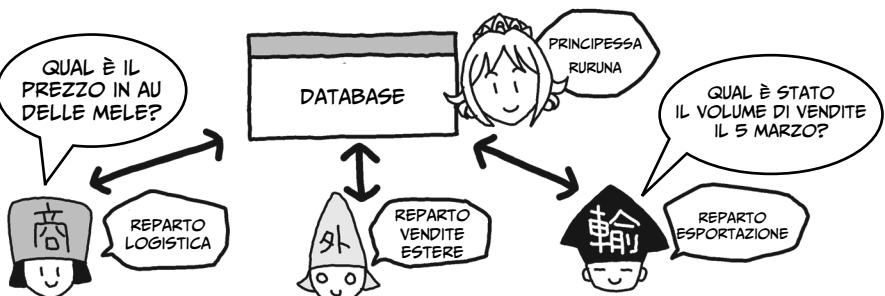
In definitiva, perché questo sistema è così inefficiente? I problemi derivano tutti dalla gestione separata e indipendente dei dati. Ma allora che dovrebbero fare Ruruna e Cain? Dovrebbero creare un database! E questo significa unificare la gestione di tutti i dati del Regno. Nel prossimo capitolo vedremo come fare.



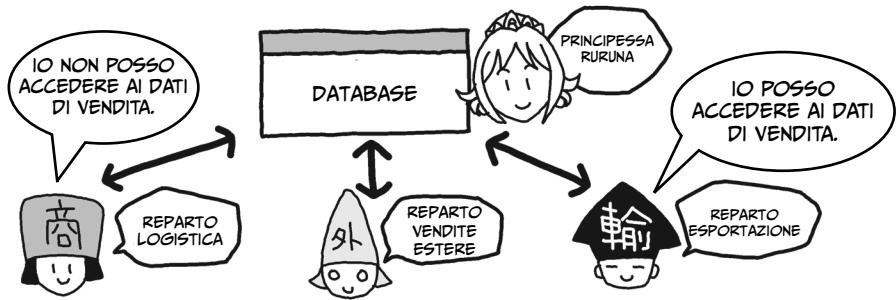
Uniformare la gestione dei dati ci assicura che ogni reparto disponga delle informazioni corrette, perché ogni reparto lancerà un'interrogazione (*query*) per ogni singolo dato. Un sistema davvero efficiente! Evita i conflitti ed elimina le duplicazioni dei dati, facilitando al contempo l'introduzione e l'integrazione di nuovi reparti.

COME SI USA UN DATABASE

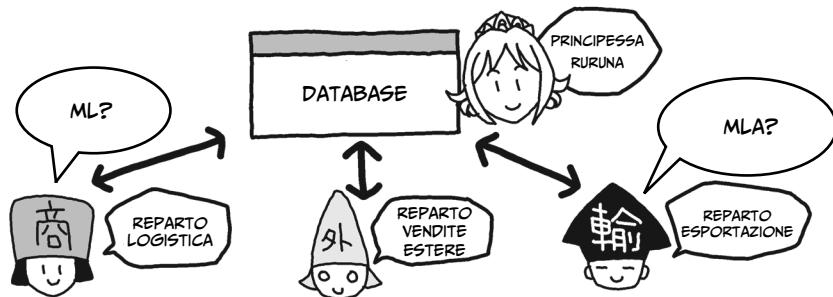
Per impostare e utilizzare un database è necessario comprendere le sue caratteristiche peculiari. Per prima cosa, un database viene usato da molte persone, e quindi occorre trovare un modo per consentire facilmente di inserire ed estrarre i dati da esso. Un metodo che sia facile e comodo per tutti.



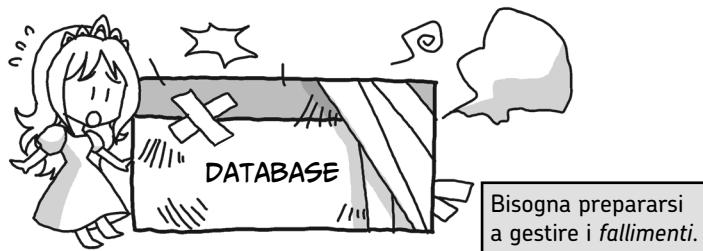
Il nuovo sistema presenta anche qualche rischio: per esempio, gli utenti potrebbero rubare o sovrascrivere informazioni importanti o confidenziali, come gli stipendi. Questi dati devono essere quindi protetti con delle restrizioni di accesso. Oppure potrebbero esserci informazioni (per esempio i dati di vendita) che devono essere accessibili solo al Reparto Esportazione. Impostare i criteri di sicurezza e i permessi è fondamentale, quando si progetta un database.



Il nuovo sistema può presentare altri problemi. Il database potrebbe essere utilizzato da molte persone contemporaneamente. Ipotizziamo che qualcuno al Reparto Vendite Estere e qualcuno al Reparto Esportazioni decidano di cambiare nello stesso istante il nome a un frutto (per esempio, il primo da *Mela* a ML e il secondo da *Mela* a MLA.) Quale sarà il nome definitivo del prodotto? Se il database verrà usato da molte persone bisogna prendere in considerazione questa eventualità.



Dovrete fare anche attenzione a non perdere nessun dato. Potrebbe anche capitare che il database vada in *crash*, o che un errore del disco rigido provochi la corruzione di qualche dato. Il database deve possedere meccanismi in grado di recuperare i dati in queste situazioni piuttosto comuni.



Inoltre, anche se il database conterrà un'enorme quantità di dati, dovremo comunque poter eseguire ricerche molto veloci. Il nuovo sistema dovrà essere in grado di gestirle.

Iniziamo lo studio dei database insieme alla Principessa Ruruna e a Cain, e impareremo a risolvere questi problemi. Andiamo al Capitolo 2!

RIASSUMENDO



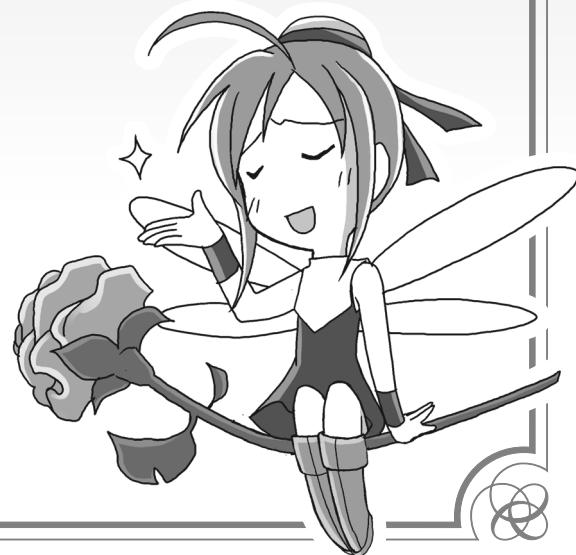
- Una gestione basata sui file può generare conflitti e una duplicazione dei dati.
- Un database ti permette di condividere facilmente i dati e di evitare conflitti e duplicazioni.

GESTIRE I DATABASE VIA SOFTWARE

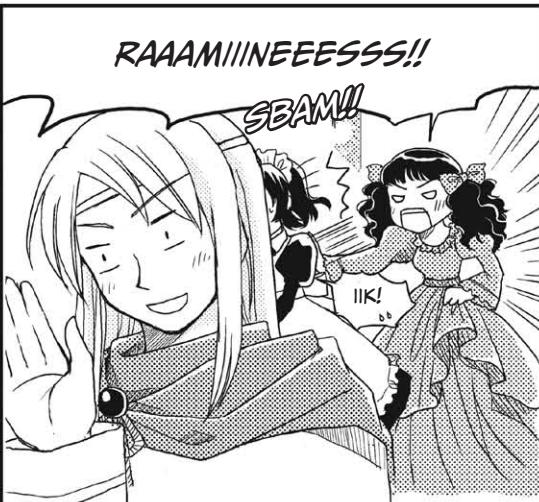
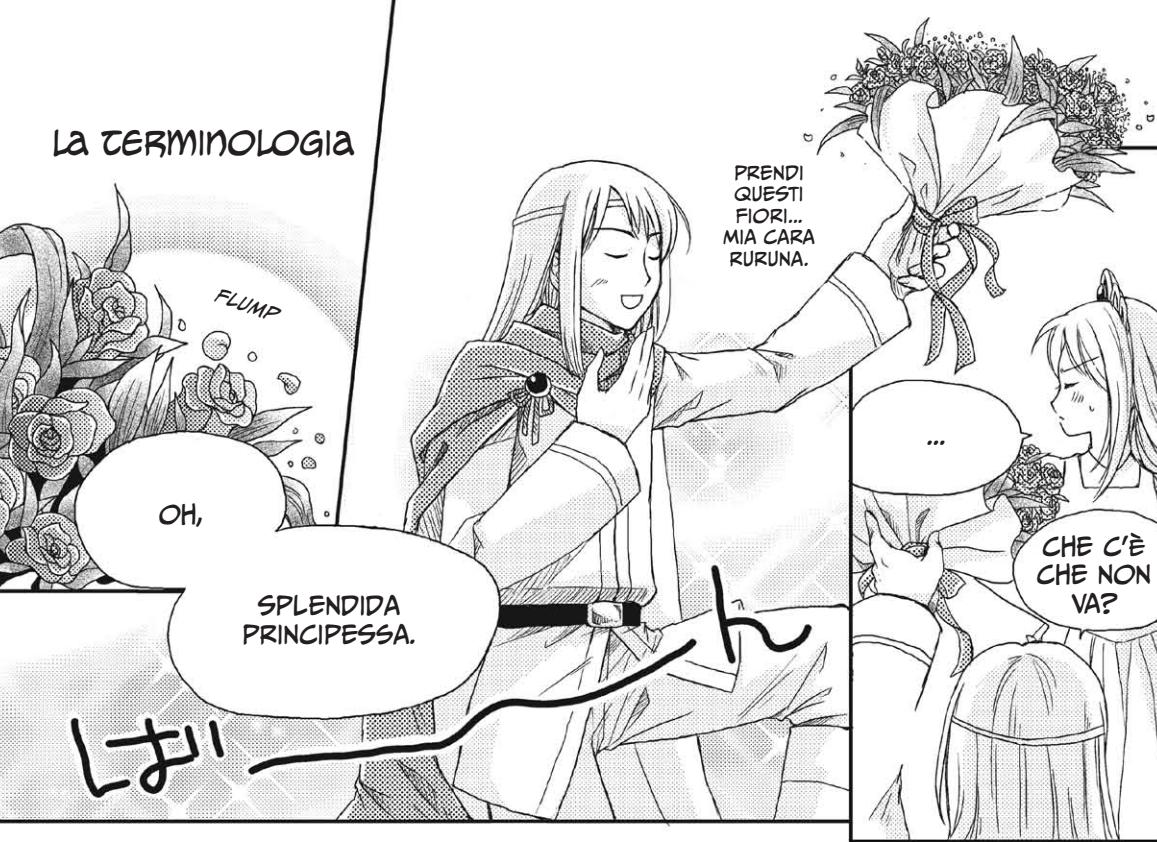
Il database che studieremo è gestito da un software chiamato *database management system* (DBMS). Un DBMS possiede molte funzioni utili e permette, tra le altre cose, di inserire dati all'interno di un database, di evitare i conflitti e di recuperare molto velocemente le informazioni. Grazie al nostro DBMS il database può essere usato da molti utenti contemporaneamente. Inoltre, il DBMS può proteggere la sicurezza del database: per esempio, consentendo al database di funzionare correttamente anche in caso di *fallimento*. Infine, il DBMS fornisce un'interfaccia semplice da usare tra il database e i suoi utenti. Studieremo i database e le funzioni di un DBMS nel prossimo capitolo.

2

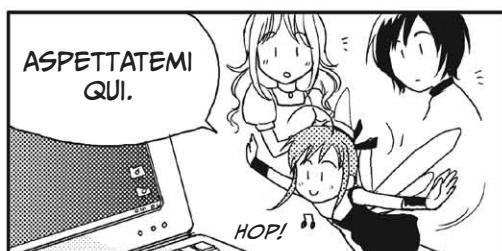
CHE COS'È UN DATABASE RELAZIONALE?



LA TERMINOLOGIA









AVETE DETTO
QUALCOSA?

N-NO, NIENTE...

È ENTRATA NEL COMPUTER!

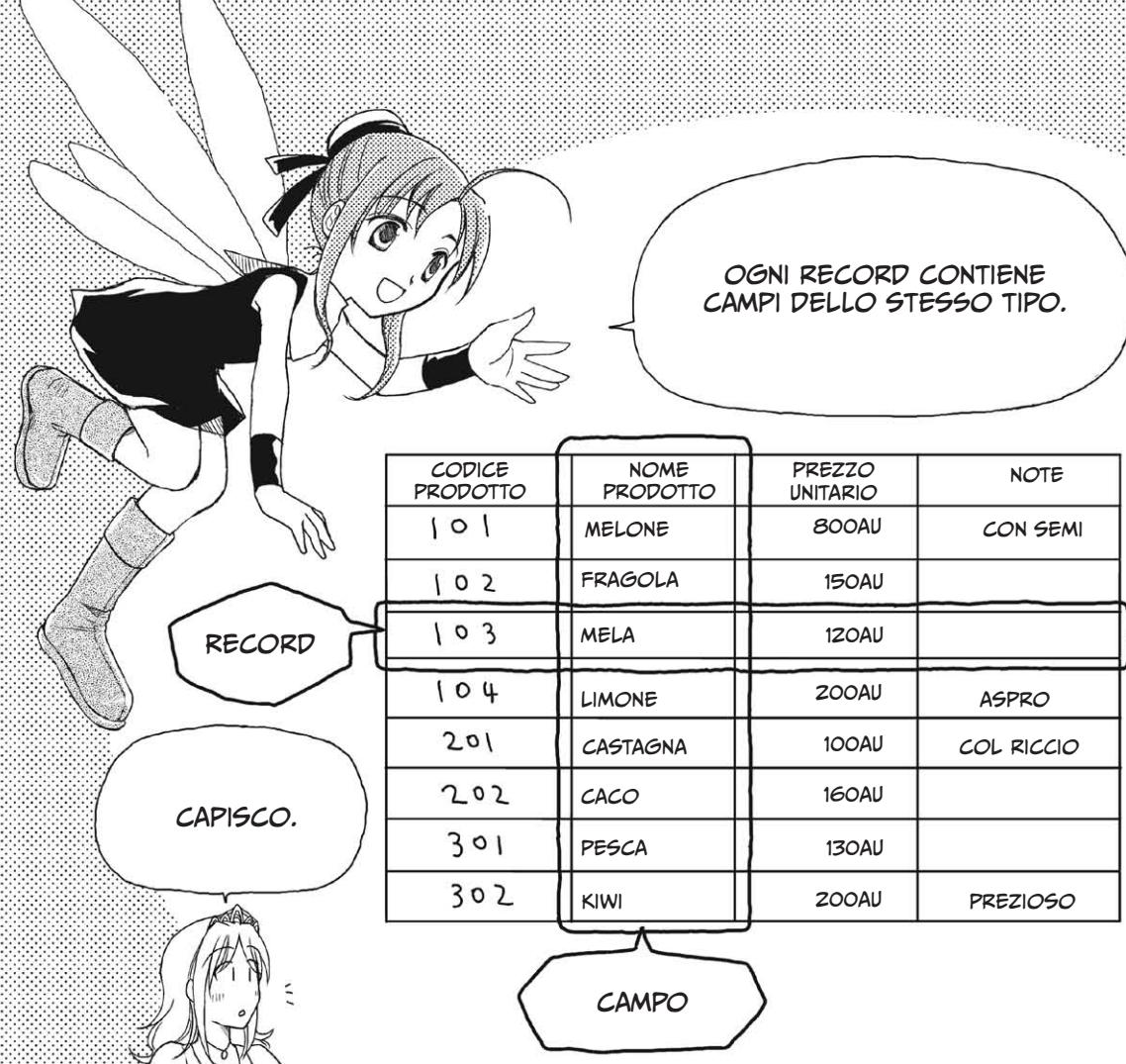
E HA TIRATO FUORI
QUALCOSA...

QUESTO È UNO
DEI FILE CHE
USATE.

UH-UH.

CODICE PRODOTTO	NOME PRODOTTO	PREZZO UNITARIO	NOTE
101	MELONE	800AU	CON SEMI
102	FRAGOLA	150AU	
103	MELA	120AU	
104	LIMONE	200AU	
201	CASTAGNA	100AU	
202	CACO	160AU	





PER ESEMPIO,
CODICE PRODOTTO È
UN NUMERO A
TRE CIFRE...

...MENTRE **NOME PRODOTTO** HA
AL MASSIMO DIECI
CARATTERI

CODICE PRODOTTO	NOME PRODOTTO
101	MELONE
102	FRAG
103	MELA
104	LIMONE
201	CASTAGNA
202	CACO

ADESSO
PROVIAMO A
CONSIDERARE
IN DETTAGLIO
IL CODICE
PRODOTTO.

CODICE PRODOTTO
101
102
103
104
201
202
301
302

HMM...
RECORD...
CAMPO...
MUMBLE...
MUMBLE...

QUI!

CAIN?



QUINDI È POSSIBILE IDENTIFICARE UN DATO GRAZIE AL CODICE PRODOTTO, MA NON CON IL PREZZO UNITARIO.

ESATTO.

NEL MONDO DEI DATABASE, UN CAMPO COME QUESTO...

...SI DEFINISCE UNIVOCO.

IL CODICE PRODOTTO
È UNIVOCO.

EQUIVOCO?

A VOLTE È QUELLO CHE DICONO DI MIO PADRE...

NO, UNIVOCO!

AH AH AH, RE KOD È UN VERO MATTACCHIONE.

VUOL DIRE CHE È UNICO ED ESCLUSIVO.

ESCLUSIVO!

UNIVOCO!
HA UN SIGNIFICATO BEN PRECISO, SAPETE?

E ORA, CONSIDERIAMO LE NOTE.

LE NOTE?

SONO COME LE CARATTERISTICHE DI UNA PERSONA, GIUSTO?

TIPO QUESTE...
CONSIDERIAMOLE DAL PUNTO DI VISTA DI UN DATABASE.

IN ALCUNI CASI DENTRO LE NOTE NON C'È SCRITTO NIENTE, GIUSTO?

PERSONA	NOTE
RURUNA	BIONDA ATTIVA
CAIN	BRUNO TRANQUI

COME, TRANQUI?



NOTE
CON SEMI
0 AU
200 AU
200 AU AMARO
100 AV CON RICCIO

CAPISCO DOVE VUOI ARRIVARE...



CODICE PRODOTTO	NOME PRODOTTO	PREZZO UNITARIO	NOTE
101	MELONE	800AU	CON SEMI
102	FRAGOLA	150AU	
103	MELA	120AU	
104	LEMON	200AU	AMARO
201	LIMONE	100AU	CON RICCIO
202	CACO	160AU	
301	PESCA	130AU	
302	KIWI	200AU	

NON PUÒ ESSERE "NULL".

NEL MONDO DEI DATABASE, QUANDO NON C'È UN VALORE SI DEFINISCE NULL.

UN VALORE NULL È ACCETTABILE DENTRO A UNA NOTA, MA NON PER UN CODICE PRODOTTO, CHE DEVE IDENTIFICARE UN DATO.



SE CONTINUI A USARE
L'ATTUALE SISTEMA DI
FILE INDIPENDENTI...



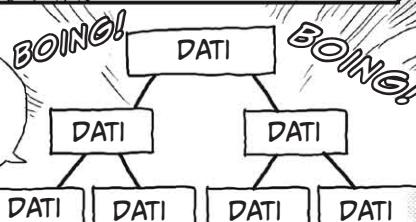
HAI RAGIONE.

È PER QUESTO CHE
VOGLIO CREARE UN
DATABASE.

E ORA DIMMI
COME SI FA!

CALMA!

PER ESEMPIO...



...QUESTO È
UN MODELLO
DI DATABASE
GERARCHICO
IN CUI...

...I DATI SONO
ORGANIZZATI SECONDO
UNA STRUTTURA AD
ALBERO.

QUANDO DICI
DATABASE, DEVI
SAPERE CHE NE
ESISTONO MOLTI
TIPI DIVERSI.

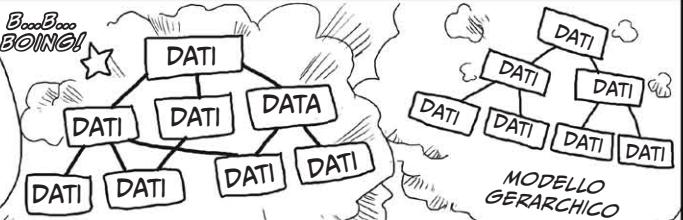


COME PER
LA FRUTTA.

DAVVERO?



POI ESISTE IL
MODELLO RETICOLARE,
IN CUI I DATI SONO
COLLEGATI L'UN L'ALTRO
TRAMITE RELAZIONI CHE
SI INTERSECANO TRA
LORO.



NON È
INCREDIBILE,
CAIN?

SONO PRONTO
A TUTTO!

B...BOING!

MODELLO
GERARCHICO

MA ALLORA
USEREMO UNO
DI QUESTI?

UMPH!

SHAZAMI



NO!

ZZZZ!

ZZB!

BANG!

BANG
BANG!

CHE, ANSIA!



IN EFFETTI C'È UN
ALTRO TIPO MOLTO
PIÙ SEMPLICE DA
USARE DI QUESTI
DUE.

PAURA,
EH?

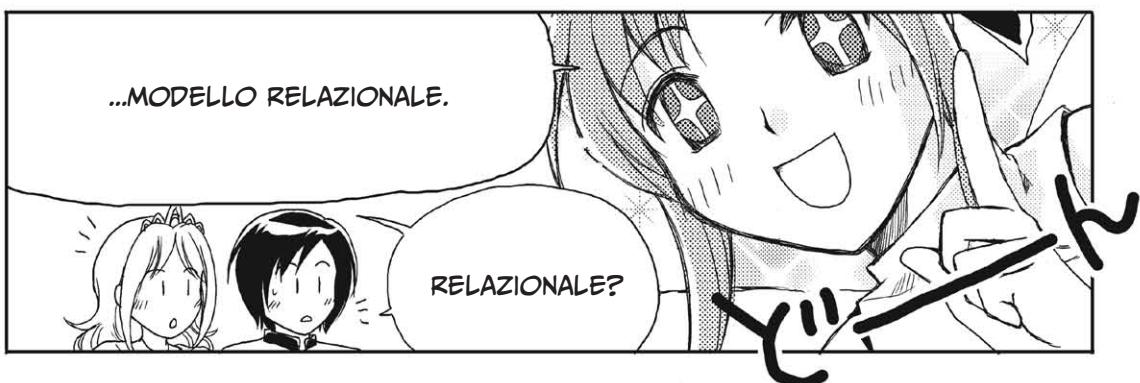
PER
NIENTE.

SI CHIAMA...

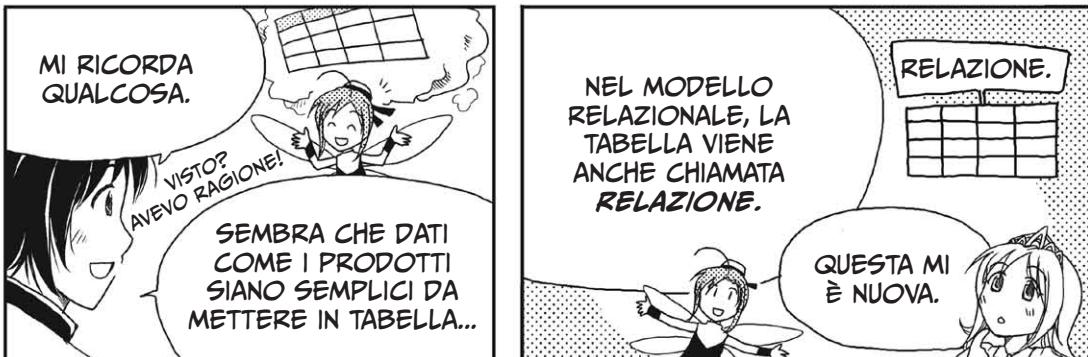
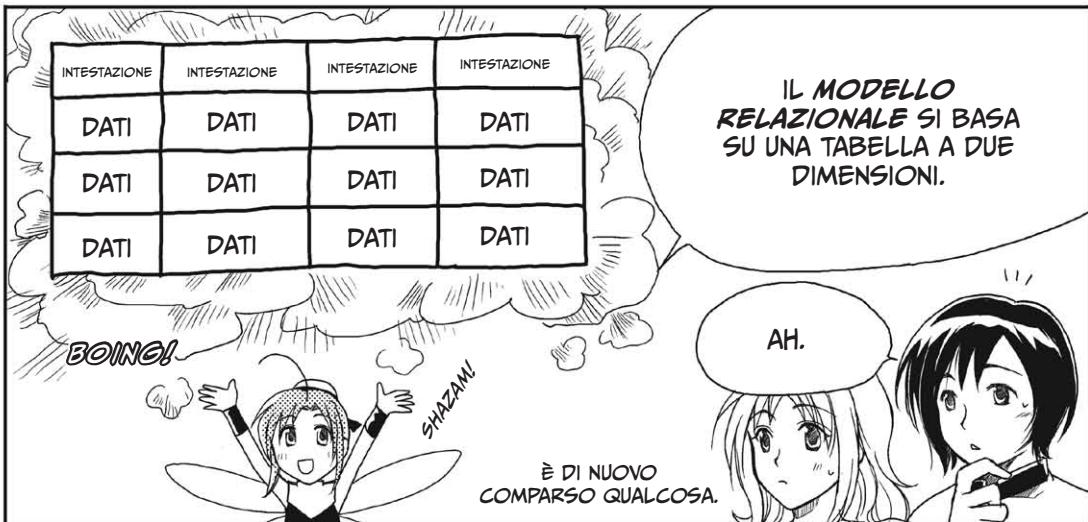


...MODELLO RELAZIONALE.

RELAZIONALE?



I DATABASE RELAZIONALI



PUÒ CAPITARE A VOLTE CHE A UN CAMPO VENGA AFFIDATO UN RUOLO MOLTO IMPORTANTE.

QUESTI CAMPI SPECIALI SONO CHIAMATI **CHIAVI**.

UN RUOLO IMPORTANTE?

SÌ, PER ESEMPIO...

...IL CODICE PRODOTTO, NEL FILE CHE HAI VISTO POCO FA.

QUEL CAMPO SVOLGE UN RUOLO IMPORTANTE, SERVE A IDENTIFICARE I DATI.

QUESTO CODICE VIENE DEFINITO **CHIAVE PRIMARIA**.

NON SAPEVO CHE CI FOSSENNO TUTTI QUESTI TERMINI.

CHIAVE PRIMARIA

CODICE PROD.
101
102
103
104
201
202
301
302

BE', IO LE TABELLE LE CONOSCO BENE.

I DATI MESSI DENTRO A UNA TABELLA SONO PIÙ COMPRENSIBILI.

È UNO DEI VANTAGGI DEL MODELLO RELAZIONALE.

ANCHE LE PERSONE CHE NON CONOSCONO I DATABASE SANNON MANEGGIARE I DATI COSÌ.

INOLTRE, IL MODELLO RELAZIONALE È COSTRUITO IN MODO DA PERMETTERE DI FARE OPERAZIONI MATEMATICHE SUI DATI.

EHM... MATEMATICHE?

LO SAPEVO, ALLA FINE È DIFFICILE...

PER NIENTE.



TORNIAMO PER ESEMPIO ALLA TABELLA DI PRIMA.

CODICE PRODOTTO	NOME PRODOTTO	PREZZO UNITARIO	NOTE
101	MELONE	800AU	CON SEMI
102	FRAGOLA	150AU	
103	MELA	120AU	
104	LIMONE	200AU	AMARO
201	CASTAGNA	100AU	CON RICCIO
202	CACO	160AU	
301	PESCA	130AU	
302	KIWI	200AU	PREZIOSO

NOME PRODOTTO
MELONE
FRAGOLA
MELA
LIMONE
CASTAGNA
CACO
PESCA
KIWI

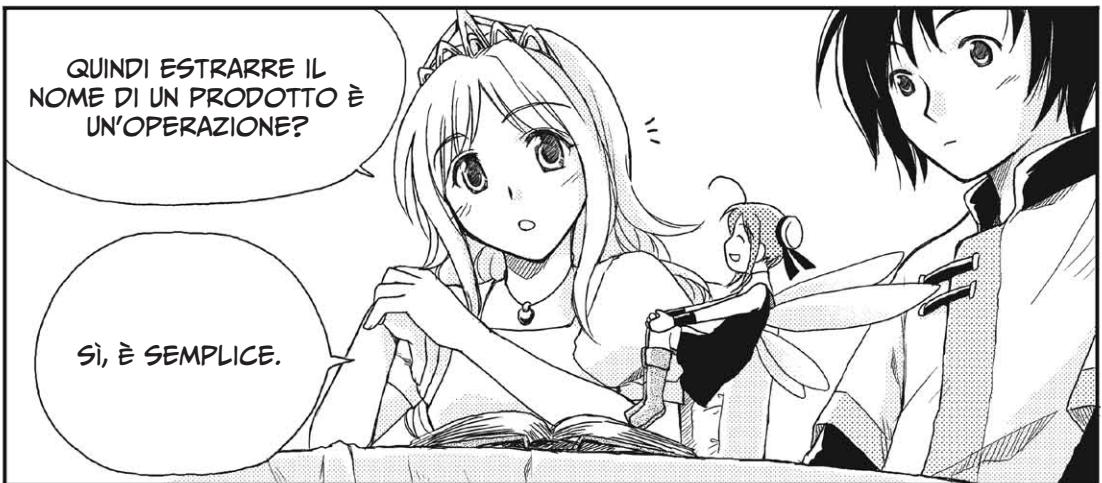
mAgia!



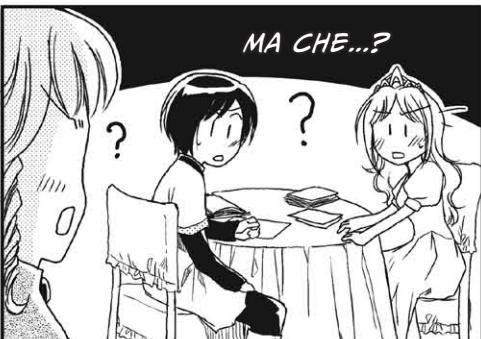
PUOI ESTRARRE IL NOME DEL PRODOTTO?

QUANDO ESTRAI UNA COLONNA FAI UN'OPERAZIONE CHIAMATA PROIEZIONE.





ALLORA CREEREMO
INSIEME UN DATABASE
RELAZIONALE PER IL
REGNO DI KOD, GIUSTO?



ESATTO!

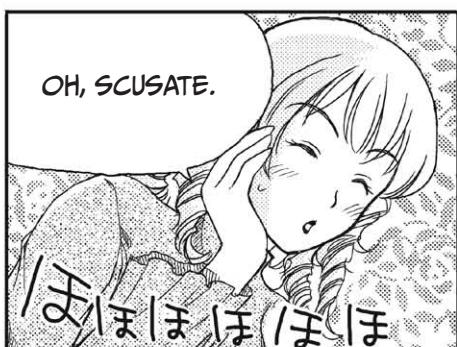
DAMMI IL
CINQUE!

IL PRINCIPE
RAMINESS È
ANDATO VIA
POCO FA...

OH!

OH, SCUSATE.

*IHIHIIHIIH



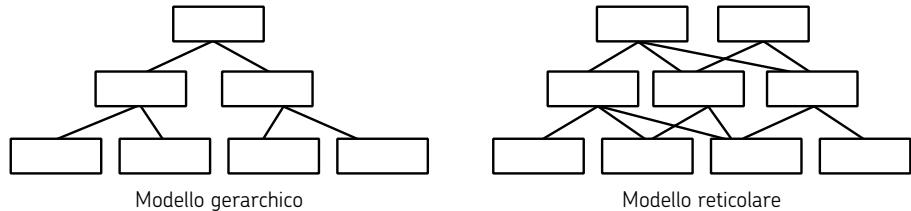
I MODELLI DI DATABASE



Quando usi il termine *database*, a quale tipo di database ti riferisci? Esistono svariate possibilità nella gestione dei dati. L'associazione dei dati e le procedure usate da un database si definiscono *modello*. I modelli più comunemente utilizzati sono tre.

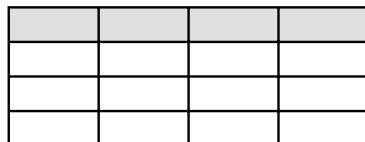
Come ho raccontato a Ruruna e a Cain, il primo tipo è il modello di database *gerarchico*. In questo modello un dato "figlio" contiene solo un pezzo del dato "genitore".

Il secondo tipo è il modello *reticolare* in cui, a differenza del modello gerarchico, un dato "figlio" può collegarsi a più di un dato "genitore".



Per usare correttamente uno di questi due modelli devi gestire i dati tenendo sempre a mente la loro allocazione fisica e il loro ordine. Quindi è complicato effettuare una ricerca flessibile e veloce dei dati usando i modelli gerarchico o reticolare.

Il terzo tipo di database si basa sul modello relazionale. Un database *relazionale* elabora i dati sfruttando un concetto semplice da comprendere, la tabella. Vediamo in dettaglio di cosa si tratta.



Modello relazionale

OPERAZIONI DI ESTRAZIONE DEI DATI

Come si estraggono i dati in un database relazionale? Puoi operare sui dati ed estrarli utilizzando operatori matematici formalmente definiti. Esistono otto operatori principali a disposizione che si dividono in due categorie: gli operatori di insieme e gli operatori relazionali.

GLI OPERATORI DI INSIEME

Unione, differenza, intersezione e prodotto cartesiano sono i quattro *operatori di insieme*.

Questi operatori manipolano una o più righe per produrre un nuovo insieme di righe. In breve, essi determinano quali righe in ingresso dovranno essere fornite in uscita.

Vediamo qualche esempio usando la Tabella Prodotti 1 e la Tabella Prodotti 2.

TABELLA PRODOTTI 1

Prodotto	Prezzo unitario
Melone	800Au
Fragola	150Au
Mela	120Au
Limone	200Au

TABELLA PRODOTTI 2

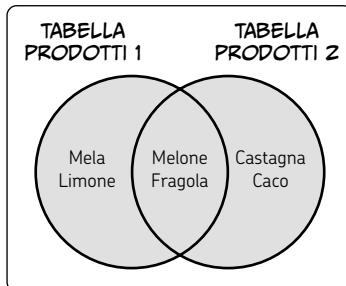
Prodotto	Prezzo unitario
Melone	800Au
Fragola	150Au
Castagna	100Au
Caco	350Au

UNIONE

L'operatore *unione* ti permette di estrarre tutti i prodotti contenuti nella Tabella Prodotti 1 e nella Tabella Prodotti 2. Questo è il risultato:

Prodotto	Prezzo unitario
Melone	800Au
Fragola	150Au
Mela	120Au
Limone	200Au
Castagna	100Au
Caco	350Au

L'operazione di unione estrae tutte le righe dalle due tabelle e le combina insieme. La figura seguente mostra come sono organizzati i dati delle due tabelle in seguito all'operazione di unione. Tutte le righe della Tabella Prodotti 1 e della Tabella Prodotti 2 sono state estratte.

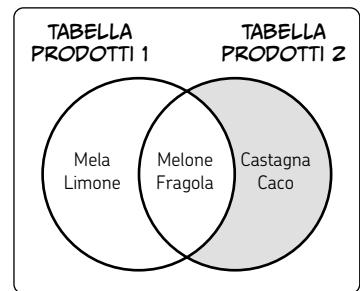
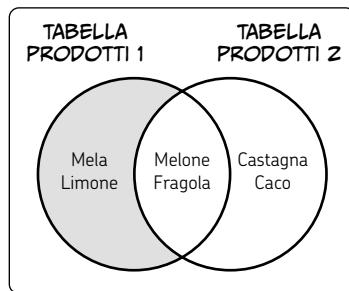


DIFFERENZA

L'operatore *differenza* estrae righe solo da *una* delle tabelle. Per esempio, un'operazione di differenza può estrarre tutti i prodotti della prima tabella che non sono inclusi nella seconda. Il risultato dipende da quale tabella si estraggono le righe e da quale tabella si escludono.

Prodotto	Prezzo unitario
Mela	120Au
Lemon	200Au

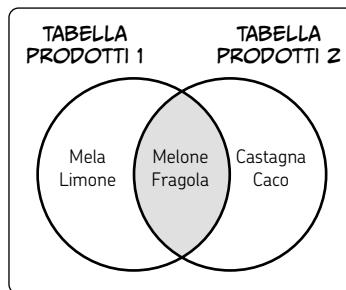
Prodotto	Prezzo unitario
Castagna	100Au
Caco	350Au



INTERSEZIONE

È anche possibile estrarre i prodotti inclusi in *entrambe* le Tabelle Prodotti. Questa operazione si chiama *intersezione*. Ecco il risultato dell'intersezione tra le Tabelle Prodotti 1 e 2.

Prodotto	Prezzo unitario
Melone	800Au
Fragola	150Au



PRODOTTO CARTESIANO

Il *prodotto cartesiano* è un'operazione che combina tutte le righe di due tabelle.

Consideriamo la Tabella Prodotti e la Tabella Destinazioni Export qui sotto.

L'operazione di prodotto cartesiano combina tutte le righe delle due tabelle. In questo esempio, le righe diventano $3 \times 3 = 9$. Osservate che i nomi delle colonne (campi) in queste due tabelle non sono gli stessi (diversamente da quanto accadeva negli esempi precedenti).

TABELLA PRODOTTI

Codice prod.	Nome prodotto	Prezzo unitario
101	Melone	800Au
102	Fragola	150Au
103	Mela	120Au

TABELLA DESTINAZIONI EXPORT

Codice dest.	Nome destinatario
12	Regno di Minanmi
23	Impero Alfa
25	Regno di Ritol

3
righe

PRODOTTO CARTESIANO

Codice prod.	Nome prodotto	Prezzo unitario	Codice dest.	Nome destinatario
101	Melone	800Au	12	Regno di Minanmi
101	Melone	800Au	23	Impero Alfa
101	Melone	800Au	25	Regno di Ritol
102	Fragola	150Au	12	Regno di Minanmi
102	Fragola	150Au	23	Impero Alfa
102	Fragola	150Au	25	Regno di Ritol
103	Mela	120Au	12	Regno di Minanmi
103	Mela	120Au	23	Impero Alfa
103	Mela	120Au	25	Regno di Ritol

3 x 3 =
9 righe



GLI OPERATORI RELAZIONALI

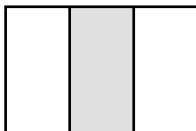
Un database relazionale è progettato in modo da permettere l'estrazione dei dati tramite gli operatori di insieme e gli operatori relazionali. Studiamo quindi gli altri quattro operatori specifici dei database relazionali. Gli *operatori relazionali* sono proiezione, selezione, congiunzione e divisione.

PROIEZIONE

L'operazione di *proiezione* estrae colonne da una tabella. Nell'esempio qui sotto, l'operazione è stata usata per estrarre solo i nomi dei prodotti inclusi nella Tabella Prodotti.

Nome prodotto
Melone
Fragola
Mela
Limone

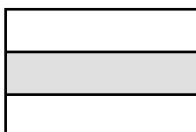
La proiezione in pratica è un'estrazione "verticale", come si vede qui sotto.



SELEZIONE

L'operazione di *selezione* può estrarre alcune righe da una tabella.

Nome prodotto	Prezzo unitario
Melone	800Au
Fragola	150Au



La selezione è come la proiezione, ma estrae le righe invece delle colonne. La selezione in pratica è un'estrazione "orizzontale."

CONGIUNZIONE

L'operatore *congiunzione* è davvero potente: infatti permette letteralmente di congiungere le tabelle. Vediamo un esempio, a partire dalle due tabelle qui sotto.

TABELLA PRODOTTI

Codice prod.	Nome prodotto	Prezzo unit.
101	Melone	800Au
102	Fragola	150Au
103	Mela	120Au
104	Limone	200Au

TABELLA VENDITE

Data	Codice prod.	Quantità
11/1	102	1.100
11/1	101	300
11/5	103	1.700
11/8	101	500

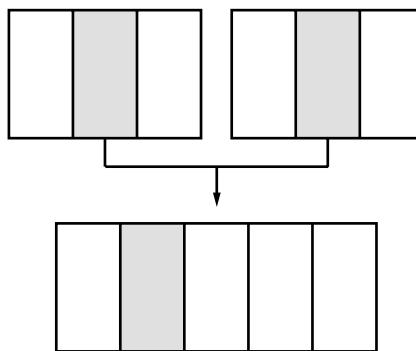
Le colonne Codice Prodotto in queste due tabelle contengono la stessa informazione.

Il primo novembre sono state vendute 1.100 fragole (codice prodotto 102). La

Tabella Vendite non contiene il nome del prodotto, ma contiene il codice prodotto. In altre parole, la Tabella Vendite permette di capire quali prodotti sono stati venduti facendo riferimento al codice prodotto, che è la *chiave primaria* nella Tabella Prodotti. Il codice prodotto nella Tabella Vendite è una *chiave esterna*. Congiungere le due tabelle, in modo che la chiave esterna riferisca alla chiave primaria, permette di ottenere la tabella seguente.

Data	Codice prod.	Nome prodotto	Prezzo unit.	Quantità
11/1	102	Fragola	150Au	1.100
11/1	101	Melone	800Au	300
11/5	103	Mela	120Au	1.700
11/8	101	Melone	800Au	500

In questo modo si crea una nuova tabella dinamica di dati di vendita, che include la data, il codice prodotto, il nome prodotto, il prezzo unitario e la quantità. La figura qui sotto mostra una congiunzione (l'area grigia indica una colonna che appare in entrambe le tabelle originali).



DIVISIONE

Per ultima c'è la divisione. La *divisione* è un'operazione che estrae righe le cui colonne corrispondono a quelle di una seconda tabella, ma restituisce esclusivamente le colonne che non compaiono nella seconda tabella. Vediamone un esempio.

TABELLA VENDITE

Codice dest.	Nome dest.	Data
12	Regno di Minanmi	3/5
12	Regno di Minanmi	3/10
23	Impero Alfa	3/5
25	Regno di Ritol	3/21
30	Regno di Sazanna	3/25

TABELLA DESTINAZIONI ESPORTAZIONE

Codice dest.	Nome dest.
12	Regno di Minanmi
23	Impero Alfa

Dividendo la Tabella Vendite per la Tabella Destinazioni Esportazione si ottiene la seguente tabella, che ti consente di trovare le date in cui la frutta è stata esportata sia verso l'Impero Alfa che verso il Regno di Minanmi.

Data
3/5

DOMANDE



E adesso rispondi a qualche domanda per verificare se hai capito come funzionano i database relazionali. Troverai le risposte a pagina 48.

D1

Come si chiama la chiave che fa riferimento a una colonna di una tabella diversa in un database relazionale?

D2

La tabella seguente propone un esempio relativo ai libri. Quale elemento puoi utilizzare come chiave primaria? Il codice ISBN (International Standard Book Number) è un numero univoco che identifica ogni libro pubblicato. Alcuni libri, invece, potrebbero avere lo stesso titolo.

ISBN	Nome libro	Autore	Data pubblicazione	Prezzo
------	------------	--------	--------------------	--------

D3

Come si chiama l'operatore utilizzato in questo caso per estrarre i dati?

Codice dest.	Nome dest.
12	Regno di Minanmi
23	Impero Alfa
25	Regno di Ritol
30	Regno di Sazanna

Codice dest.	Nome dest.
25	Regno di Ritol

D4

Come si chiama l'operatore utilizzato in questo caso per estrarre i dati?

Codice dest.	Nome dest.
12	Regno di Minanmi
23	Impero Alfa
25	Regno di Ritol
30	Regno di Sazanna

Codice dest.	Nome dest.
15	Regno di Paronu
22	Regno di Tokanta
31	Regno di Taharu
33	Regno di Mariyon



Codice dest.	Nome dest.
12	Regno di Minanmi
15	Regno di Paronu
22	Regno di Tokanta
23	Impero Alfa
25	Regno di Ritol
30	Regno di Sazanna
31	Regno di Taharu
33	Regno di Mariyon

D5

Come si chiama l'operatore utilizzato in questo caso per estrarre i dati?

Codice dest.	Nome dest.
12	Regno di Minanmi
23	Impero Alfa
25	Regno di Ritol
30	Regno di Sazanna

Codice dest.	Data
12	3/1
23	3/1
12	3/3
30	3/5
12	3/6
25	3/10



Codice dest.	Data	Nome dest.
12	3/1	Regno di Minanmi
23	3/1	Impero Alfa
12	3/3	Regno di Minanmi
30	3/5	Regno di Sazanna
12	3/6	Regno di Minanmi
25	3/10	Regno di Ritol

IL DATABASE RELAZIONALE VINCE!

In un database relazionale puoi usare fino a otto operatori diversi per estrarre i dati. I risultati sono forniti sotto forma di tabella. Combinando le operazioni che hai imparato a conoscere in questa sezione potrai ottenere qualsiasi tipo di dato. Per esempio, puoi usare il nome e il prezzo di un prodotto per creare una tabella aggregata con i risultati di vendita. I database relazionali sono molto diffusi perché sono semplici da capire e molto flessibili nella gestione dei dati.

Riassumendo



- Una riga di dati si chiama record, mentre le colonne si chiamano campi.
- Una colonna che può essere usata per identificare i dati viene chiamata chiave primaria.
- In un database relazionale puoi elaborare i dati usando il concetto di tabella.
- In un database relazionale puoi elaborare i dati utilizzando delle operazioni matematiche.

RISPOSTE

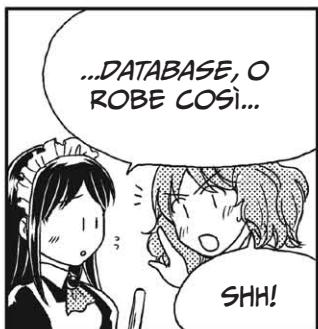
- D1** Chiave Esterna.
D2 ISBN.
D3 Selezione.
D4 Unione.
D5 Congiunzione.

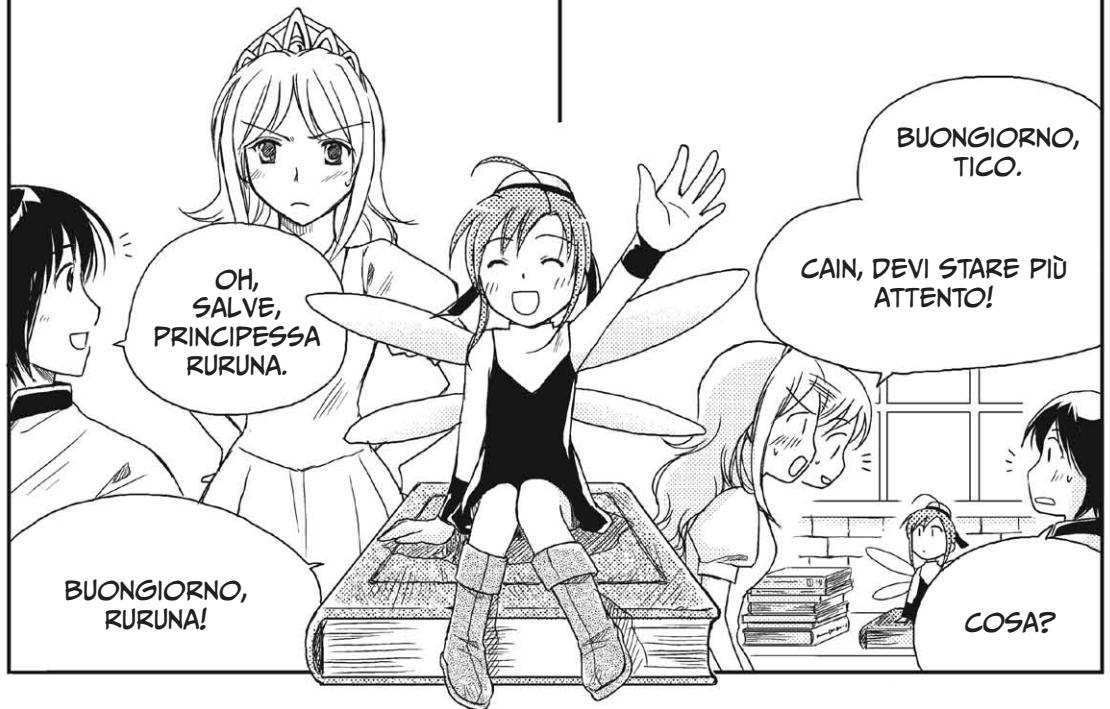
3

PROGETTIAMO UN DATABASE!

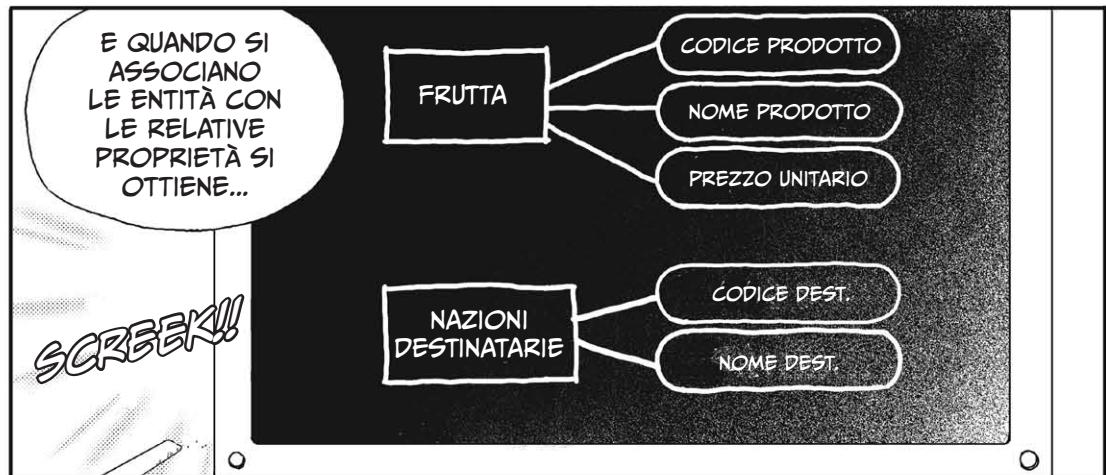


IL MODELLO E-R

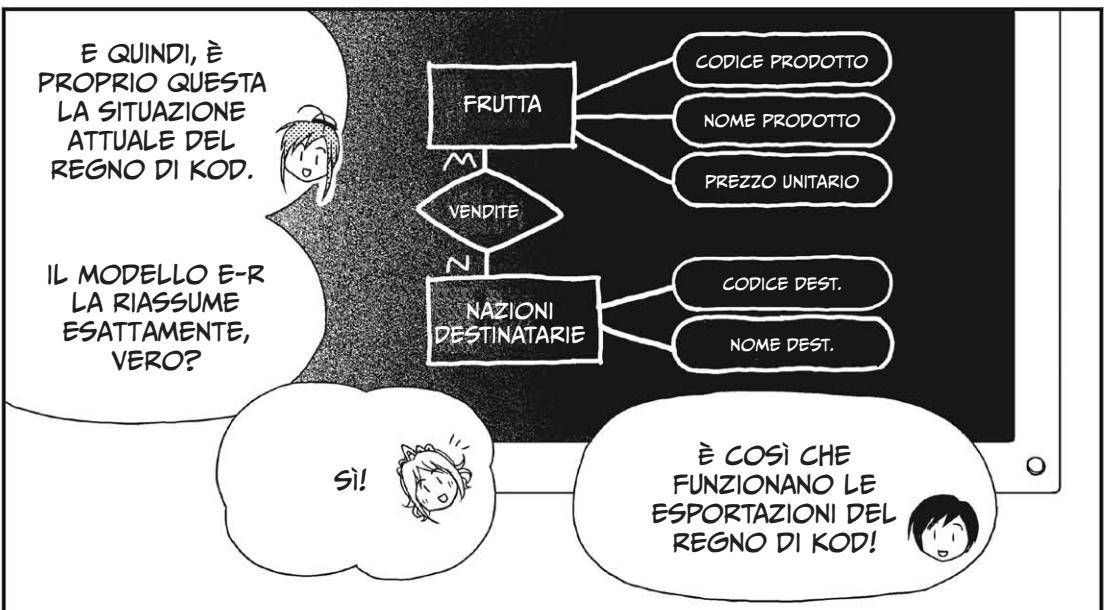
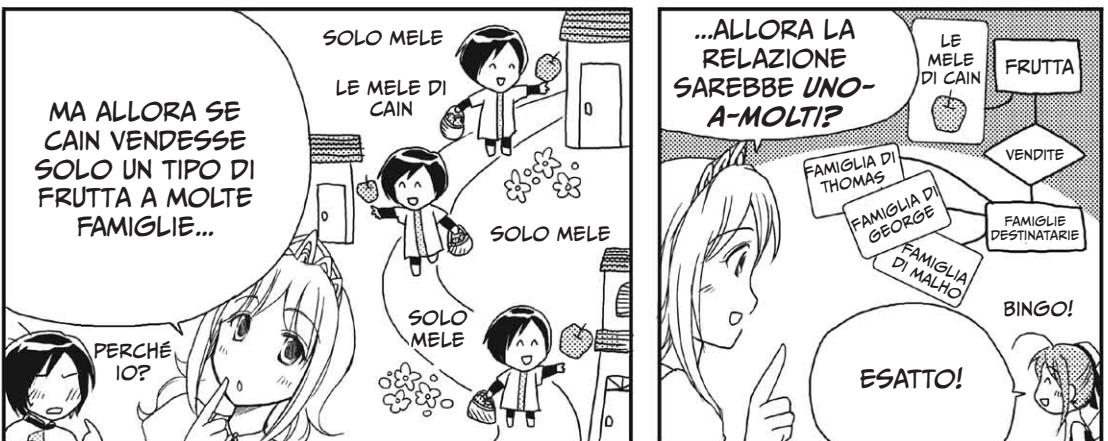
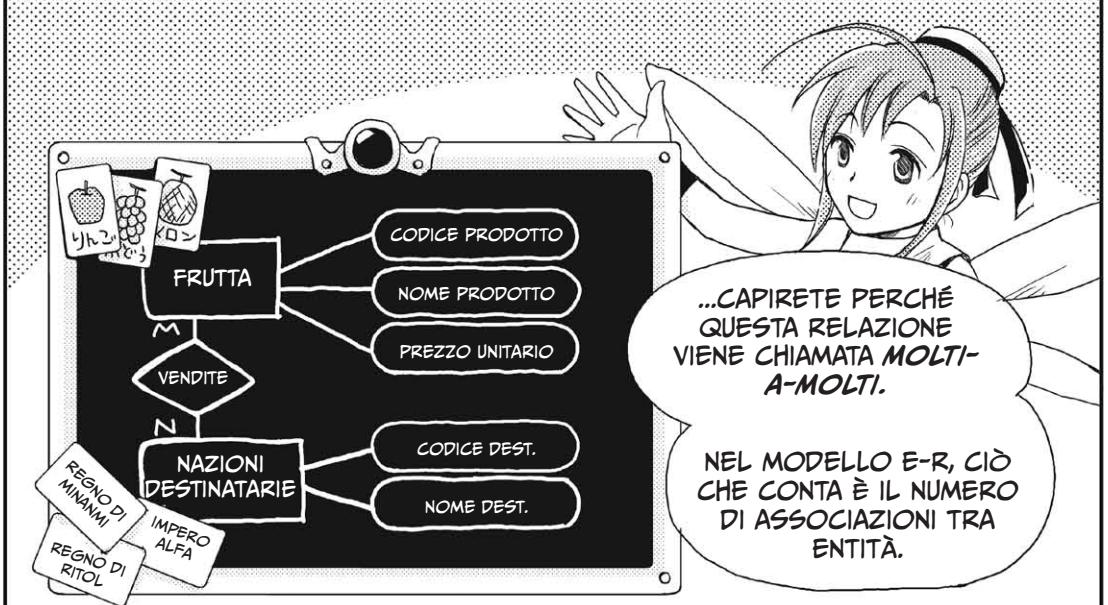












NORMALIZZARE UNA TABELLA

È DAVVERO DIFFICILE COMINCIARE A CREARE UN DATABASE.

VERO! LA PRIMA COSA DA FARE È ANALIZZARE LA SITUAZIONE DI PARTENZA. È MOLTO IMPORTANTE.

E ORA CHE AVETE DEFINITO LA SITUAZIONE ATTUALE DEL REGNO DI KOD...

...POSSIAMO COMINCIARE A PROGETTARE UN VERO DATABASE.

S!!!!!!

SIGNOR CAIN?

GASP

??

SHHH!

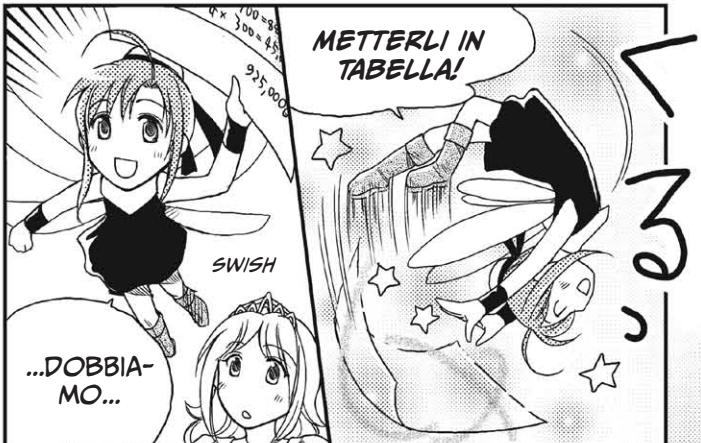
PRINCIPESSA?

MM...

TROVATO.

MI TUFFO!

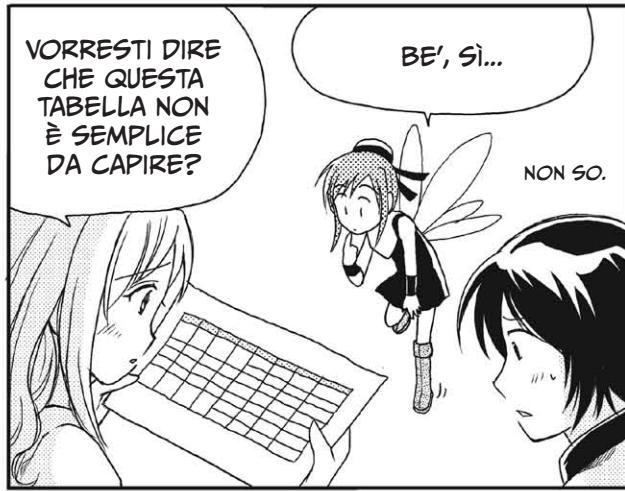
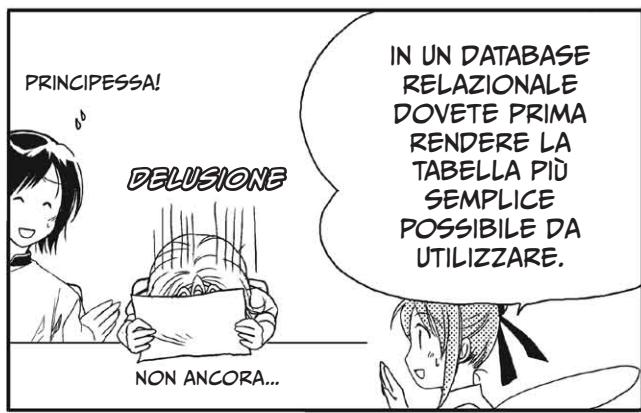
REGISTRAZIONE DI VENDITA	
1101 AL REGNO DI MINAMI	DATA: 3/5
101 MELONE @ 800AU X 1, 100=880, 000AU	
102 FRAGOLA @ 150AU X 300= 45, 000AU	
REGNO DI KOD	TOTALE 925, 000AU

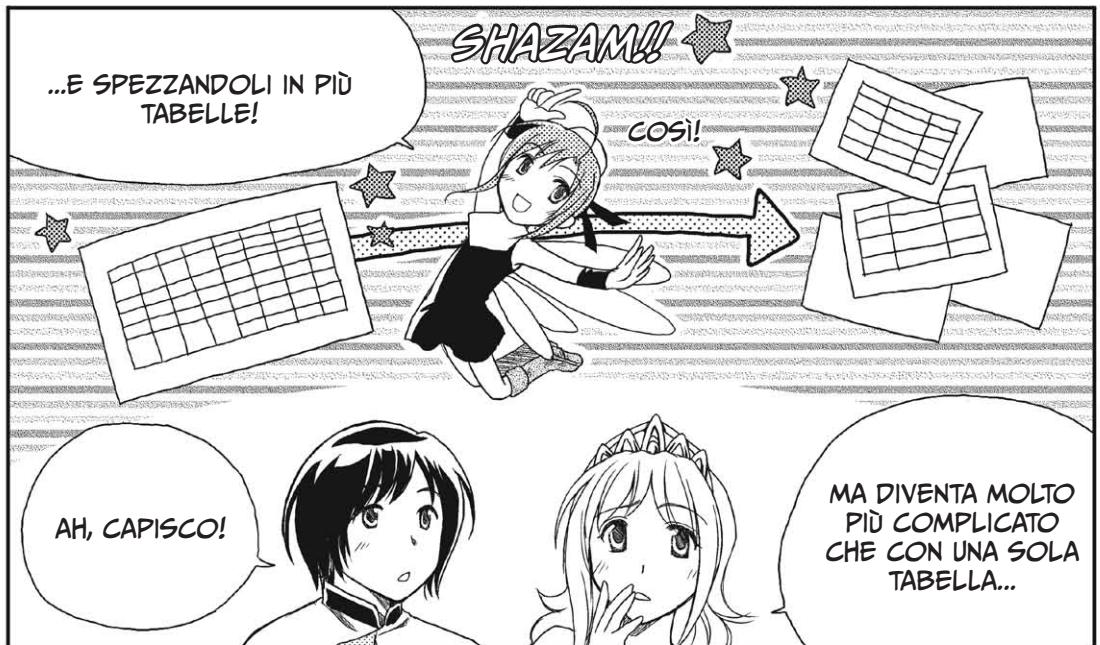
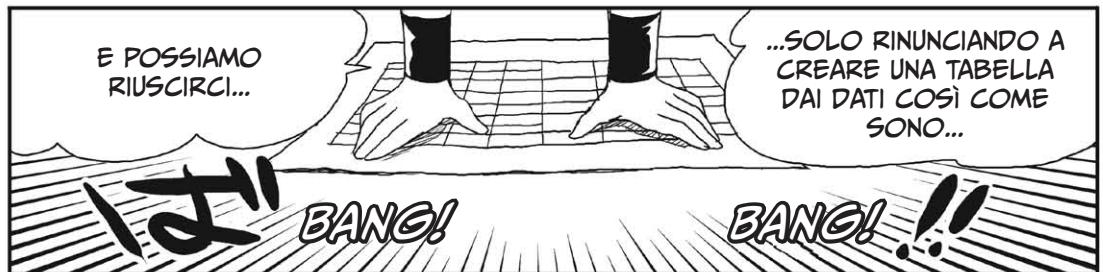


ECCOLA QUI.

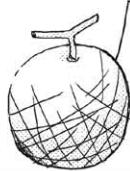
N. REGISTRAZIONE	DATA	CODICE DEST.	NOME DEST.	CODICE PRODOTTO	NOME PRODOTTO	PREZZO UNIT.	QUANTITÀ
1101	3/5	12	REGNO DI MINANMI	101	MELONE	800AU	1.100
				102	FRAGOLA	150AU	300
1102	3/7	23	IMPERO ALFA	103	MELA	120AU	1.700
1103	3/8	25	REGNO DI RITOL	104	LIMONE	200AU	500
1104	3/10	12	REGNO DI MINANMI	101	MELONE	800AU	2.500
1105	3/12	25	REGNO DI RITOL	103	MELA	120AU	2.000
				104	LIMONE	200AU	700

TABELLA CREATA A PARTIRE DAI RAPPORTI DI VENDITA





IPOZZIAMO DI VOLER AUMENTARE IL PREZZO DEL MELONE DI ZOAU.



20AU



SE USASTE LA TABELLA COSÌ COM'È...

...DOVRESTE TROVARE TUTTE LE RIGHE CHE CONTENGONO MELONI E CORREGGERE I PREZZI UNO ALLA VOLTA.

820AU

800AU

150AU

E ANCHE QUI...

00AU

820AU

0AU

È QUI!

MA CON UNA TABELLA CHE CONTIENE SOLO I SINGOLI PRODOTTI...

...VI BASTERÀ CORREGGERE IL PREZZO UNA VOLTA SOLA, UN SINGOLO RECORD DELLA TABELLA PRODOTTI.

TABELLA PRODOTTI	
MELONE	800AU
FRAGOLA	150AU
MELA	120AU
LIMONE	200AU

SOLO QUI!

820AU

FACILE!

E SENZA ALCUN RISCHIO DI CREARE CONFLITTI, ANCHE SE VI SCORDATE DI CORREGGERE GLI ALTRI RECORD! NON È FENOMENALE?

SE USI UNA SOLA TABELLA È FACILE CHE TI DIMENTICHI DI CORREGGERE QUALCOSA.

CAPISCO, SOTTO QUESTA LUCE NON È VANTAGGIOSA.



UH...

DIVIDERE LA TABELLA PER EVITARE POSSIBILI CONFLITTI TRA I DATI...

...SI CHIAMA FARE UNA NORMALIZZAZIONE.

NORMALIZZAZIONE, NORMALIZZAZIONE...

È MOLTO IMPORTANTE!

HISS

PARLI ANCORA DA SOLO?

MA ALLORA COSA DOVREI FARE?

INNANZI-TUTTO...



...IN UNA TABELLA CHE CONTIENE LA DATA, IL CODICE DESTINATARIO E IL NOME DEL DESTINATARIO...

...E IN UN'ALTRA TABELLA CHE CONTIENE CODICE PRODOTTO, NOME DEL PRODOTTO, PREZZO UNITARIO E QUANTITÀ.

TABELLA VENDITE (PRIMA FORMA NORMALE (1))

N. REGISTRAZIONE	DATA	CODICE DEST.	NOME DEST.
1101	3/5	12	REGNO DI MINANMI
1102	3/7	23	IMPERO ALFA
1103	3/8	25	REGNO DI RITOL
1104	3/10	12	REGNO DI MINANMI
1105	3/12	25	REGNO DI RITOL

MA IL NUMERO DI REGISTRAZIONE COMPARTE IN ENTRAMBE LE TABELLE, È GIUSTO?

TABELLA VENDITE (PRIMA FORMA NORMALE (2))

N. REGISTRAZIONE	CODICE PRODOTTO	NOME PRODOTTO	PREZZO UNIT.	QUANTITÀ
1101	101	MELONE	800AU	1.100
1101	102	FRAGOLA	150AU	300
1102	103	MELA	120AU	1.700
1103	104	LIMONE	200AU	500
1104	101	MELONE	800AU	2.500
1105	103	MELA	120AU	2.000
1105	104	LIMONE	200AU	700

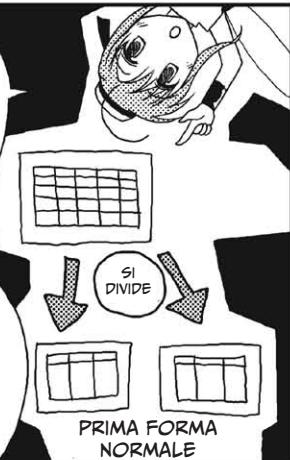


LA TABELLA CHE RISULTA DA QUESTA DIVISIONE PRENDE IL NOME DI **PRIMA FORMA NORMALE**.



LA TABELLA CHE INVECE HA RECORD CHE CONTENGONO DUE O PIÙ VALORI, PRIMA DI VENIRE DIVISA VIENE CHIAMATA **FORMA NON NORMALIZZATA**.

IN PRATICA LA PRIMA FORMA NORMALE SI OTTIENE DIVIDENDO LA FORMA NON NORMALIZZATA.



VEDIAMO...

...UN MOMENTO.

IL FATTO CHE QUESTE SIANO "PRIME FORME NORMALI" LASCA INTENDERE CHE ESISTANO ANCHE "SECONDE" E "TERZE" FORME NORMALI, GIUSTO?

ESATTO!

IN UN DATABASE RELAZIONALE NON SI PUÒ USARE UNA PRIMA FORMA NORMALE COSÌ COM'È.

MONTE DATABASE RELAZIONALE

CORAGGIO!

AH, CAPISCO...

RESISTI!

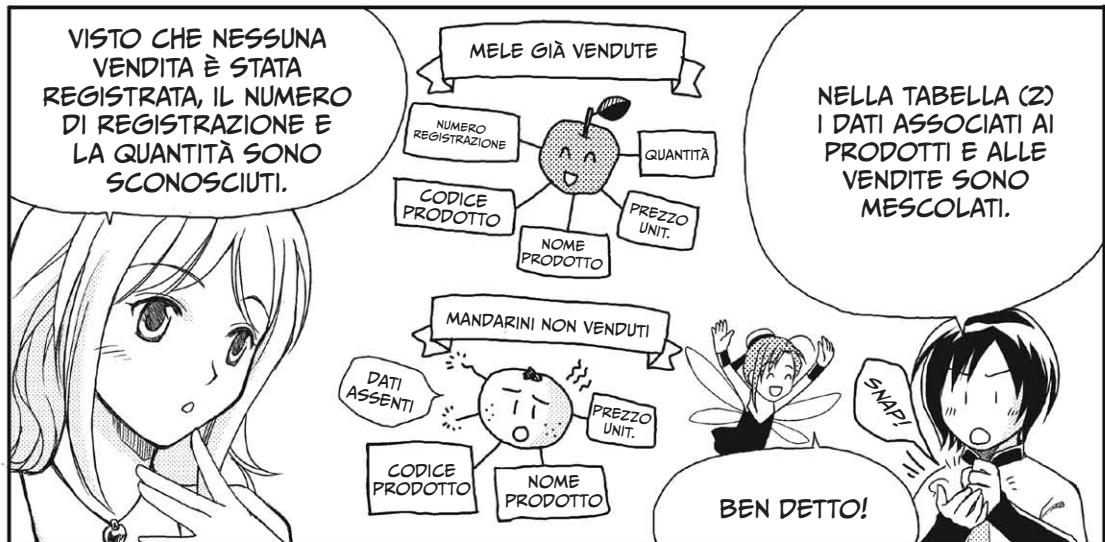
MA QUANDO FINISCE?!

PUFF PANT



N. REGISTRAZIONE	CODICE PRODOTTO	NOME PRODOTTO	PREZZO UNIT.	QUANTITÀ
1101	101	MELONE	800AU	1.100
1101	102	FRAGOLA	150AU	300

TABELLA RISULTATI DI VENDITA
(PRIMA FORMA NORMALE (2))





**TABELLA PRODOTTI
(SECONDA FORMA NORMALE (1))**

CODICE PRODOTTO	NOME PRODOTTO	PREZZO UNIT.
101	MELONE	800AU
102	FRAGOLA	150AU
103	MELA	120AU
104	LIMONE	200AU



LA TABELLA (1) CONTIENE I DATI RELATIVI AI PRODOTTI.

SE NELLA COLONNA DEL CODICE PRODOTTO È ASSEGNAZIONATO UN VALORE POSSIAMO TROVARE IL NOME PRODOTTO E IL PREZZO UNITARIO CORRISPONDENTI.

**TABELLA PRODOTTI
(SECONDA FORMA NORMALE (2))**

N. REGISTRAZIONE	CODICE PRODOTTO	QUANTITÀ
1101	101	1.100
1101	102	300
1102	103	1.700
1103	104	500
1104	101	2.500
1105	103	2.000
1105	104	700

QUINDI IL CODICE PRODOTTO È LA CHIAVE PRIMARIA E DETERMINA I VALORI DELLE ALTRE COLONNE.

CAVOLI.

ESATTO.

PER QUANTO
RIGUARDA I DATI
RELATIVI ALLE
VENDITE NELLA
TABELLA (2),

IN QUESTA
TABELLA, LA
CHIAVE PRIMARIA
DETERMINA I
VALORI NELLE
ALTRE COLONNE.

MA...

IMMAGINIAMO NELLA
TABELLA (2) DI USARE
COME CHIAVE PRIMARIA LA
COMBINAZIONE DI NUMERO
DI REGISTRAZIONE E
CODICE PRODOTTO.

CHIAVE PRIMARIA

N. REGISTRAZIONE	CODICE PRODOTTO

CI SONO CASI
IN CUI VENDIAMO
CONTEMPORANEAMENTE
DUE PRODOTTI
DIVERSI...

E CI SONO CASI IN CUI
VENDIAMO QUANTITÀ
DIVERSE DELLO STESSO
PRODOTTO.

QUESTO SIGNIFICA...

...CHE SI DIVIDE LA
TABELLA FINCHÉ, UNA
VOLTA DETERMINATA LA
CHIAVE PRIMARIA, QUESTA
DETERMINA I VALORI DELLE
ALTRE COLONNE.

CAPITO?

①

CODICE PRODOTTO	NOME PRODOTTO	PREZZO UNITARIO
-----------------	---------------	-----------------

CHIAVE PRIMARIA

②

N. REGISTRAZIONE	CODICE PRODOTTO	QUANTITÀ
------------------	-----------------	----------

CHIAVE PRIMARIA

CAPITO.

LA TABELLA OTTENUTA DALLA DIVISIONE SECONDO QUESTA REGOLA SI CHIAMA...

SECONDA FORMA NORMALE.

NELLA SECONDA FORMA NORMALE (1) POSSIAMO AGGIUNGERE I MANDARINI DI CUI PARLAVAMO PRIMA.

POSSIAMO AGGIUNGERE ANCHE L'UVA E I KIWI,

CHE NON SONO STATI ANCORA VENDUTI!

E SE IL PREZZO DEL MELONE CAMBIA, È SUFFICIENTE MODIFICARE IL VALORE IN UN SINGOLO RECORD, GIUSTO?

...A PROPOSITO, HAI DIVISO LA PRIMA FORMA NORMALE (2)...

UH?

...MA ALLORA NON È PIÙ NECESSARIO DIVIDERE LA TABELLA DELLA PRIMA FORMA NORMALE (1)?

OH, PORTI GLI OCCHIALI, ADESSO.

FLUMP

VEDIAMO!

OTTIMA OSSERVAZIONE!

TABELLA VENDITE
(PRIMA FORMA NORMALE (1))

N. REGISTRAZIONE	DATA	CODICE DEST.	NOME DEST.
1101	3/5	12	REGNO DI MINANMI
1102	3/7	23	IMPERO ALFA
1103	3/8	25	REGNO DI RITOL
1104	3/10	12	REGNO DI MINANMI
1105	3/12	25	REGNO DI RITOL



IN QUESTA TABELLA, SE UN NUMERO DI REGISTRAZIONE È DETERMINATO, TUTTI GLI ALTRI VALORI (DATA, CODICE DESTINATARIO E NOME DESTINATARIO) SONO DETERMINATI.

S!!!



TABELLA VENDITE (PRIMA FORMA NORMALE (1))				TABELLA VENDITE (SECONDA FORMA NORMALE (3))			
N. REGISTRAZIONE	DATA	CODICE DEST.	NOME DEST.	N. REGISTRAZIONE	DATA	CODICE DEST.	NOME DEST.
		REGNO DI MINANMI				REGNO DI MINANMI	
		IMPERO ALFA				IMPERO ALFA	
		REGNO DI RITOL				REGNO DI RITOL	

GIUSTO. PUOI
CONSIDERARE LA PRIMA
FORMA NORMALE (1)...
...COME LA SECONDA
FORMA NORMALE (3)!



CONSIDERIAMO DI NUOVO LA SECONDA FORMA NORMALE (3).



NON PUOI GESTIRE LE DESTINAZIONI DELLE TUE ESPORTAZIONI CON QUESTA TABELLA.

PENSA,
PENSA,
PENSA...
AH!

TABELLA VENDITE
SECONDA FORMA NORMALE (3)

N. REGISTRAZIONE	DATA	CODICE DEST.	NOME DEST.
1101	3/5	12	REGNO DI MINANMI
	3/7	23	IMPERO ALFA
	3/8	25	REGNO DI RITOL

IL REGNO DI SAZANNA, A CUI NON SONO STATE FATTE ESPORTAZIONI FINORA, NON PUÒ ESSERE AGGIUNTO A QUESTA TABELLA.



NELLA TABELLA (3) I DATI RELATIVI ALLE DESTINAZIONI E ALLE VENDITE SONO MISCHIATI.

MM...

COME FACCIAMO A GESTIRE IN MANIERA AUTONOMA LE DESTINAZIONI?

...DIVIDIAMO DI NUOVO!

TABELLA VENDITE
TERZA FORMA NORMALE (1)

N. REGISTRAZIONE	DATA	CODICE DEST.
1101	3/5	12
1102	3/7	23
1103	3/8	25
1104	3/10	12
1105	3/12	25

TABELLA DESTINAZIONI TERZA FORMA NORMALE (2)

CODICE DEST.	NOME DEST.
12	REGNO DI MINANMI
23	IMPERO ALFA
25	REGNO DI RITOL

PROPRIO COSÌ...

SHAZAM!

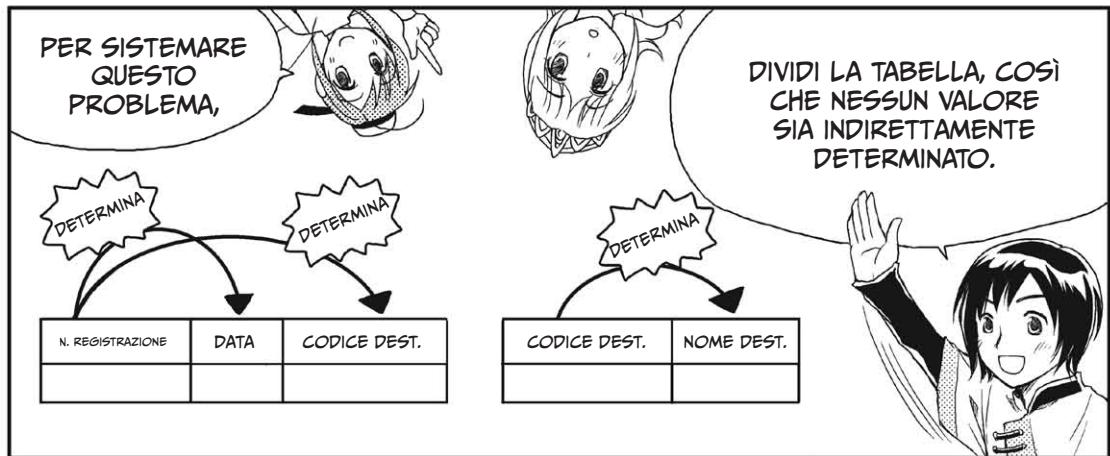


TABELLA VENDITE

N. REGISTRAZIONE	DATA	CODICE DEST.
1101	3/5	12
1102	3/7	23
1103	3/8	25
1104	3/10	12
1105	3/12	25

TABELLA DESTINAZIONI

CODICE DEST.	NOME DEST.
12	REGNO DI MINANMI
23	IMPERO ALFA
25	REGNO DI RITOL

TABELLA RISULTATI DI VENDITA

N. REGISTRAZIONE	CODICE PRODOTTO	QUANTITÀ
1101	101	1.100
1101	102	300
1102	103	1.700
1103	104	500
1104	101	2.500
1105	103	2.000
1105	104	700

TABELLA PRODOTTI

CODICE PRODOTTO	NOME PRODOTTO	PREZZO UNITARIO
101	MELONE	800AU
102	FRAGOLA	150AU
103	MELA	120AU
104	LIMONE	200AU

UN DATABASE RELAZIONALE UTILIZZA NORMALMENTE TABELLE DIVISE FINO ALLA TERZA FORMA NORMALE.



QUESTE SONO LE TABELLE CHE SI OTTENGONO DIVIDENDO UNA TABELLA FINO ALLA TERZA FORMA NORMALE.

ORA IL NOSTRO DATABASE È COMPLETO!

EVVÀ!



ORA POTETE GESTIRE I PRODOTTI, LE DESTINAZIONI E LE VENDITE PASSANDO DA UNA TABELLA ALL'ALTRA...

...SENZA PROBLEMI.

PRODOTTO

DESTINAZIONE

VENDITE

GIÀ GIÀ.

ANCHE AGGIUNGERE NUOVI DATI NON CREERÀ CONFLITTI.

CHE SOLLIEVO...

ANCHE SE ABBIAMO DIVISO LA TABELLA ORIGINALE IN QUATTRO TABELLE AGGIUNTIVE,

IN QUESTE TABELLE HANNO TROVATO POSTO TUTTI I DATI ORIGINALI.

TABELLA RISULTATI DI VENDITA

TABELLA DESTINAZIONI

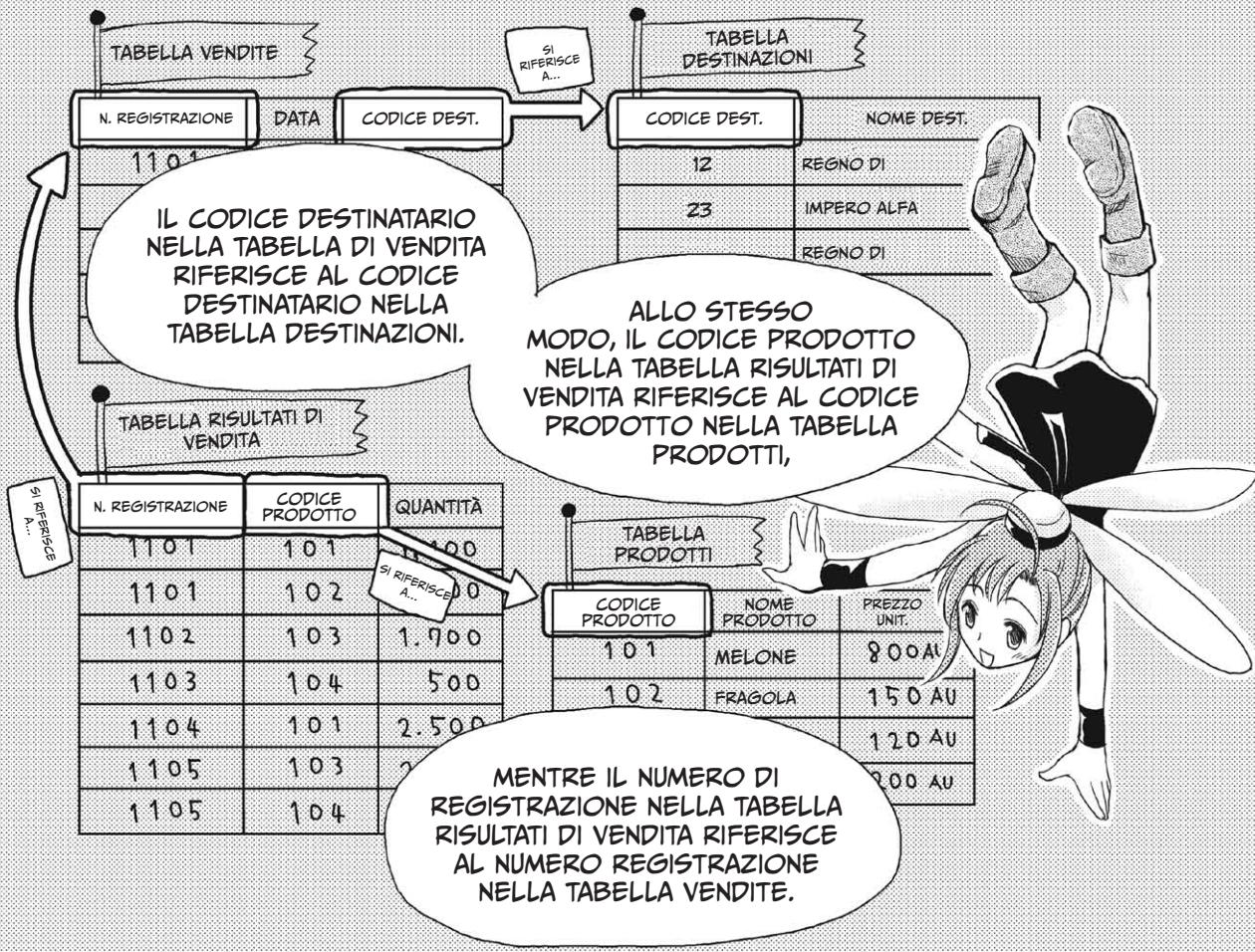
TABELLA VENDITE

TABELLA PRODOTTI

TABELLA CREATI DAI RISULTATI DI VENDITA

NELLA PROSSIMA PAGINA SI VEDONO ESPlicitate le RELAZIONI TRA I DATI.

ESATTO! È UN DATABASE RELAZIONALE.



NON VEDO L'ORA
DI POTER GESTIRE
PIÙ FACILMENTE LE
ESPORTAZIONI GRAZIE
AL NOSTRO DATABASE.

STANCO

Sì.

OH, SEI
TU.

QUALCOSA
NON VA?

PRINCIPESSA...
MR CAIN...

VOI DUE VI STATE
COMPORTANDO IN
MODO DAVVERO
STRANO.

CHE VI
PRENDE?

ASCOLTA,
POSso SPIEGARTI
TUTTO.

NON AVrà
DETTO
QUALCOSA DI
STRANO ALLA
PRINCIPESSA,
SIGNOR CAIN!

IH IH
IO? CERTO
CHE NO.

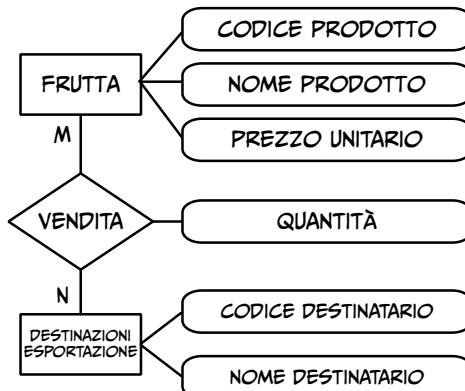
CHE COS'È IL MODELLO E-R?



Cain e la Principessa Ruruna hanno analizzato la situazione attuale del Regno di Kod usando un modello E-R (entità-relazione). Quando proverai a creare un database, il primo passo sarà determinare le condizioni dei dati che stai cercando di modellare.

Cerca quindi di definire un'entità nei tuoi dati, usando il modello E-R. L'entità è un oggetto concreto o una cosa realmente esistente, come la *frutta* o le *destinazioni* verso cui esporti.

In aggiunta, un modello E-R evidenzia le relazioni tra le entità. La Principessa Ruruna e Cain hanno basato la loro analisi sulla relazione chiamata *vendita* tra la frutta e le sue destinazioni. La frutta viene esportata verso varie destinazioni, mentre ognuna di queste destinazioni importa tipi diversi di frutta. Per questa ragione l'analisi secondo il modello E-R presume che esista una relazione chiamata *molti-a-molti* tra la frutta e le sue destinazioni. M frutti hanno relazioni con N destinazioni. Il numero di associazioni tra le entità si chiama *cardinalità*.

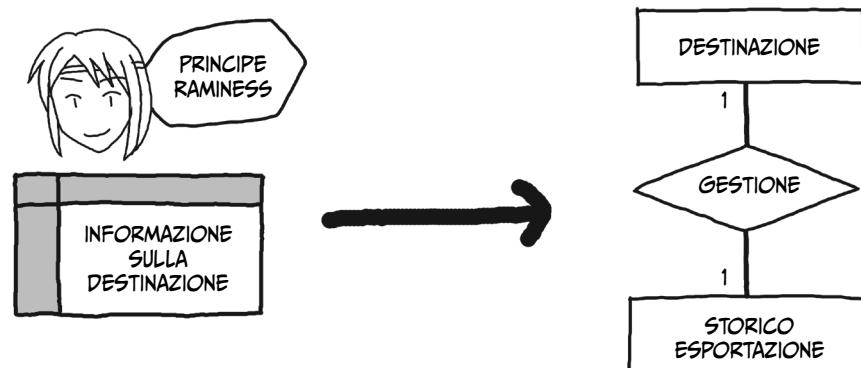


COME SI ANALIZZA IL MODELLO E-R

Prova a riflettere su come procederesti nei casi seguenti:

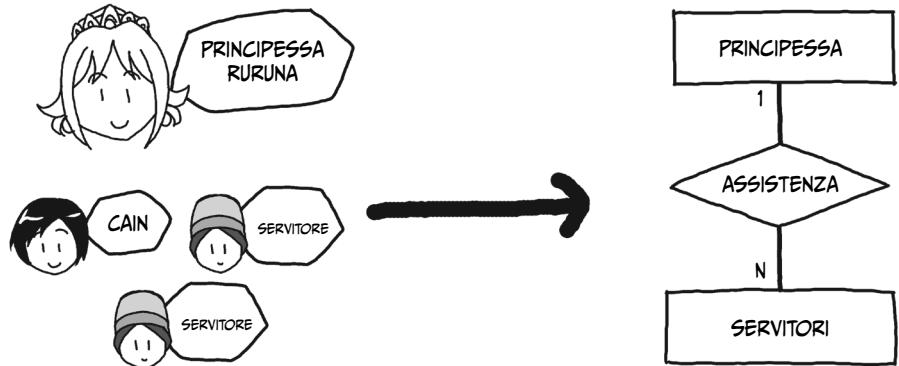
CASO 1: ASSOCIAZIONI UNO-A-UNO

Una destinazione gestisce una singola informazione di vendita. Questo tipo di relazione viene chiamata associazione *uno-a-uno*.



CASO 2: ASSOCIAZIONI UNO-A-MOLTI

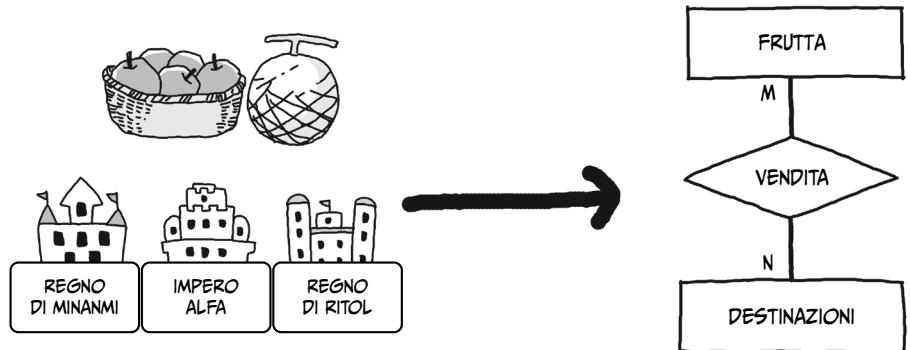
Molte persone sono al servizio di una principessa. I servitori non assistono nessun'altra principessa e neppure il re.



Questo tipo di relazione viene chiamata associazione *uno-a-molti*.

CASO 3: ASSOCIAZIONI MOLTI-A-MOLTI

La frutta viene esportata verso molti paesi. Le destinazioni importano più di un tipo di frutta.



Questo tipo di relazione viene chiamata associazione *molti-a-molti*.

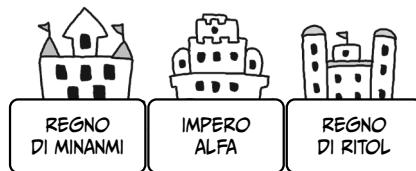


DOMANDE

Hai capito bene come funziona il modello E-R? Analizza e disegna il modello E-R per ognuno dei seguenti casi. Troverai le risposte a pagina 82.

D1

Un membro del personale gestisce diversi clienti. Un cliente non sarà mai contattato da più di un membro del personale.



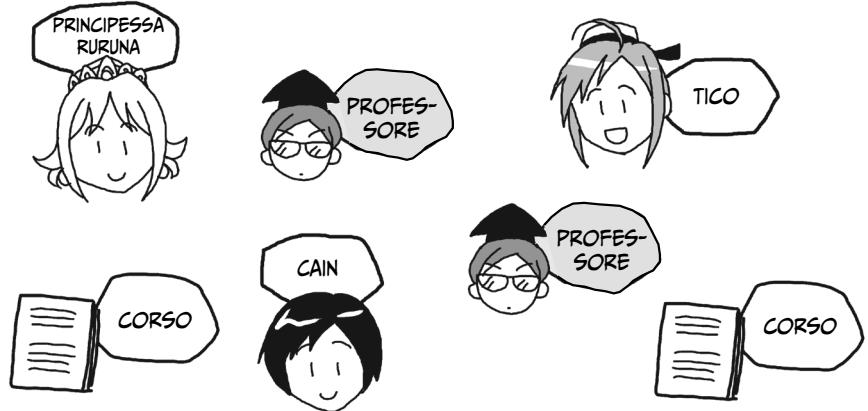
D2

Una persona può prendere in prestito più di un libro. I libri possono essere presi in prestito da più studenti in momenti diversi.

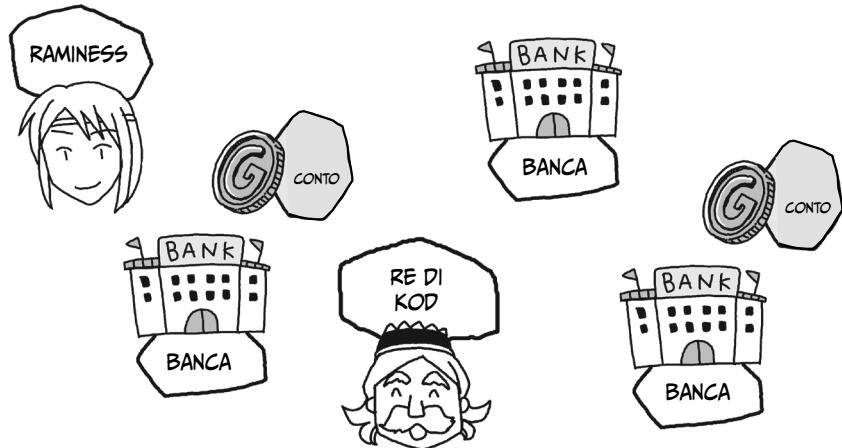


D3

Ogni studente frequenta più di un corso. Ogni corso è frequentato da più studenti. Un professore può tenere più corsi. Ogni corso è tenuto da un solo professore.

**D4**

Ogni cliente può aprire più di un conto corrente. Ogni conto corrente è aperto da un singolo cliente. Ogni banca gestisce diversi conti correnti. Ogni conto corrente è gestito da una banca.



Ricordate sempre che l'analisi basata sul modello E-R non produce necessariamente un solo risultato "corretto". Possono esserci molti modi di organizzare logicamente dei dati per riflettere le condizioni attuali di un contesto.

NORMALIZZARE UNA TABELLA



Cain e la Principessa Ruruna hanno imparato a usare la normalizzazione, il processo che consente di disporre in una tabella dei dati provenienti dal mondo reale, azione necessaria per creare un database relazionale. La normalizzazione viene riassunta qui (i campi in grigio rappresentano le *chiavi primarie*).

FORMA NON NORMALIZZATA

N. Registrazione	Data	Codice destinatario	Nome destinatario	Codice prodotto	Nome prodotto	Prezzo unitario	Quantità
------------------	------	---------------------	-------------------	-----------------	---------------	-----------------	----------

PRIMA FORMA NORMALE

N. Registrazione	Data	Codice destinatario	Nome destinatario
N. Registrazione	Codice prodotto	Nome prodotto	Prezzo unitario

SECONDA FORMA NORMALE

N. Registrazione	Data	Codice destinatario	Nome destinatario
N. Registrazione		Codice prodotto	Quantità
Codice prodotto	Nome prodotto	Prezzo unitario	

TERZA FORMA NORMALE

N. Registrazione	Data	Codice destinatario
Codice destinatario		Nome destinatario
N. Registrazione	Codice prodotto	Quantità
Codice prodotto	Nome prodotto	Prezzo unitario

La *forma non normalizzata* è una tabella che contiene elementi che compaiono più di una volta. Abbiamo visto che non è possibile gestire bene i dati usando questo tipo di tabella in un database relazionale, quindi è necessario suddividerla.

La *prima forma normale* si riferisce a una semplice tabella bidimensionale che risulta dalla divisione della tabella originaria nella forma non normalizzata. Si tratta di una tabella che contiene un elemento in ogni cella. La tabella viene divisa in modo che un elemento non compaia più di una volta.

La *seconda forma normale* si riferisce a una tabella in cui una chiave che può identificare i dati determina i valori nelle altre colonne. Stiamo parlando della *chiave primaria*.

In un database relazionale, un valore è definito *funzionalmente dipendente* se quel valore determina i valori di altre colonne. Nella seconda forma normale, la tabella è divisa per far sì che le colonne siano funzionalmente dipendenti dalla chiave primaria.

Nella *terza forma normale*, la tabella è divisa per far sì che un valore non sia determinato da una chiave che non sia primaria. In un database relazionale, un valore è definito *transitivamente dipendente* se quel valore determina indirettamente i valori di altre colonne, come parte di un'operazione funzionalmente dipendente. Nella terza forma normale la tabella è divisa in modo da poter rimuovere i valori transitivamente dipendenti.

DOMANDE



È importante riuscire a progettare un database relazionale nelle situazioni più disparate, quindi esaminiamo alcuni esempi di tabelle da normalizzare. Determina il livello di normalizzazione in tutti i casi seguenti. Troverai le risposte a pagina 82.

D5

La seguente tabella gestisce il prestito di libri (come definito nella domanda D2). Fino a quale livello è stata normalizzata?

Codice prestito	Data	Codice studente	Nome studente	Indirizzo studente	Dipartimento	Anno iscrizione
-----------------	------	-----------------	---------------	--------------------	--------------	-----------------

ISBN	Titolo	Autore	Data pubblicazione	Pagine totali
------	--------	--------	--------------------	---------------

Codice prestito	ISBN	Quantità
-----------------	------	----------

D6

Anche la seguente tabella descrive la gestione di una biblioteca. A quale livello è normalizzata?

Codice prestito	Data	Codice studente
-----------------	------	-----------------

Codice studente	Nome studente	Indirizzo studente	Dipartimento	Anno iscrizione
-----------------	---------------	--------------------	--------------	-----------------

ISBN	Titolo	Autore	Data pubblicazione	Pagine totali
------	--------	--------	--------------------	---------------

Codice prestito	ISBN	Quantità
-----------------	------	----------

D7

La tabella seguente mostra i risultati delle vendite effettuate mensilmente da ciascun dipendente. Ogni reparto è formato da più dipendenti. Un dipendente può far parte di un solo reparto. Normalizza questa tabella fino alla terza forma normale.

Codice dipendente	Nome dipendente	Mese	Vendite attribuite	Codice reparto	Nome reparto
-------------------	-----------------	------	--------------------	----------------	--------------



D8

La tabella seguente rappresenta un sistema per la ricezione degli ordini. Normalizzala fino alla terza forma normale. Ipotizza che a ogni numero d'ordine corrisponda un solo cliente. Puoi processare anche più di un prodotto con lo stesso numero d'ordine. Infine, a un numero d'ordine può essere associato solo un venditore.

Numero d'ordine	Data	Codice cliente	Nome cliente	Codice prodotto	Nome prodotto	Prezzo unitario	Codice venditore	Nome venditore	Quantità
-----------------	------	----------------	--------------	-----------------	---------------	-----------------	------------------	----------------	----------

D9

La tabella seguente rappresenta un sistema per la ricezione degli ordini. Normalizzala fino alla terza forma normale. Ipotizza che i prodotti siano classificati per codice prodotto.

Numero d'ordine	Data	Codice cliente	Nome cliente	Codice prodotto	Nome prodotto	Prezzo unitario	Codice categoria prodotto	Nome categoria prodotto	Quantità
-----------------	------	----------------	--------------	-----------------	---------------	-----------------	---------------------------	-------------------------	----------

I PASSI PER PROGETTARE UN DATABASE

Hai imparato come si progetta un database! Tuttavia questo non basta, avrai bisogno di progettare una struttura dettagliata dei file che formeranno il database e stabilire i metodi per l'esportazione e l'importazione dei dati. In generale, è possibile dividere la progettazione di un database in tre parti: lo schema concettuale, lo schema interno e lo schema esterno.

Lo *schema concettuale* si riferisce alla modellizzazione del mondo reale. Più precisamente, è un modo per determinare la struttura logica di un database. Lo schema concettuale viene progettato prendendo in considerazione la concezione del mondo basata sul modello E-R e la normalizzazione di una tabella.

Lo *schema interno* si riferisce al database visto nell'ottica di un computer. Nello specifico, è un modo per determinare la struttura fisica di un database. Lo schema interno viene progettato dopo che si è creato un metodo di ricerca ad alta velocità dei dati contenuti nel database.

Lo *schema esterno* si riferisce al database visto nell'ottica dei suoi utenti o delle applicazioni che si interfacciano con lui. Lo schema esterno viene progettato dopo che sono stati creati i dati necessari per i programmi applicativi.



SCHEMA INTERNO

SCHEMA CONCETTUALE

SCHEMA ESTERNO

In questo capitolo, la Principessa Ruruna e Cain hanno progettato un database orientato allo schema concettuale. Stanno ancora lavorando all'ottimizzazione del loro database.

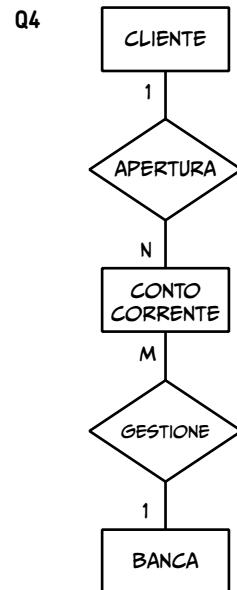
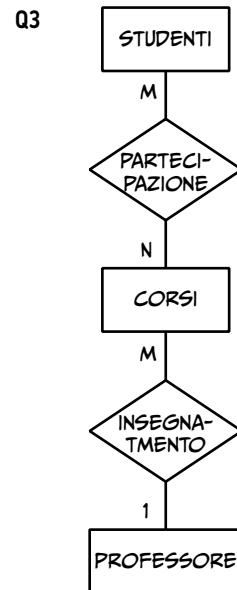
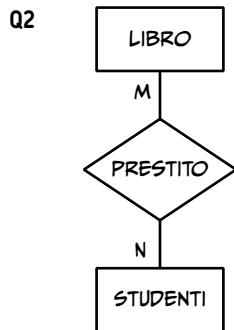
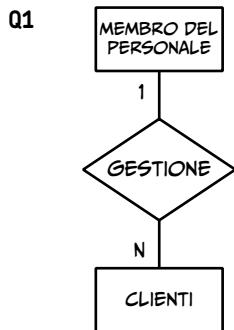
Ora che hai imparato le nozioni di base della progettazione di un database, nel prossimo capitolo vedremo concretamente come utilizzarlo.

RiASSUMENDO



- Un modello E-R è usato per analizzare le entità e le relazioni.
- Le relazioni tra le entità possono essere di tipo uno-a-uno, uno-a-molti e molti-a-molti.
- I dati in una tabella devono essere normalizzati prima di poter essere usati per creare un database relazionale.
- La progettazione di un database si può dividere in tre parti: lo schema concettuale, lo schema interno e lo schema esterno.

RISPOSTE



D5 Seconda forma normale

D6 Terza forma normale

D7

Codice dipendente	Mese	Vendite
-------------------	------	---------

Codice dipendente	Nome dipendente	Codice reparto
-------------------	-----------------	----------------

Codice reparto	Nome reparto
----------------	--------------

D8

Codice ordine	Data	Codice cliente	Codice venditore
---------------	------	----------------	------------------

Codice cliente	Nome cliente
----------------	--------------

Codice ordine	Codice prodotto	Quantità
---------------	-----------------	----------

Codice prodotto	Nome prodotto	Prezzo unitario
-----------------	---------------	-----------------

Codice venditore	Nome venditore
------------------	----------------

D9

Codice ordine	Data	Codice cliente
---------------	------	----------------

Codice cliente	Nome cliente
----------------	--------------

Codice ordine	Codice prodotto	Quantità
---------------	-----------------	----------

Codice prodotto	Codice categoria prodotto	Nome prodotto	Prezzo unitario
-----------------	---------------------------	---------------	-----------------

Codice categoria prodotto	Nome categoria prodotto
---------------------------	-------------------------

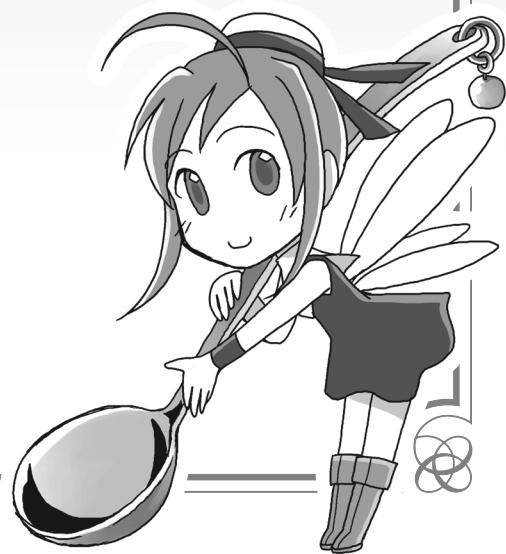
PROGETTARE UN DATABASE

In questo capitolo hai imparato come si progetta un database relazionale. Esistono tuttavia altri metodi di progettazione. L'efficienza e la praticità d'uso di un database dipendono dall'analisi e dal metodo di progettazione, quindi è importante creare un database adatto alle proprie esigenze sin dalla fase di progettazione.

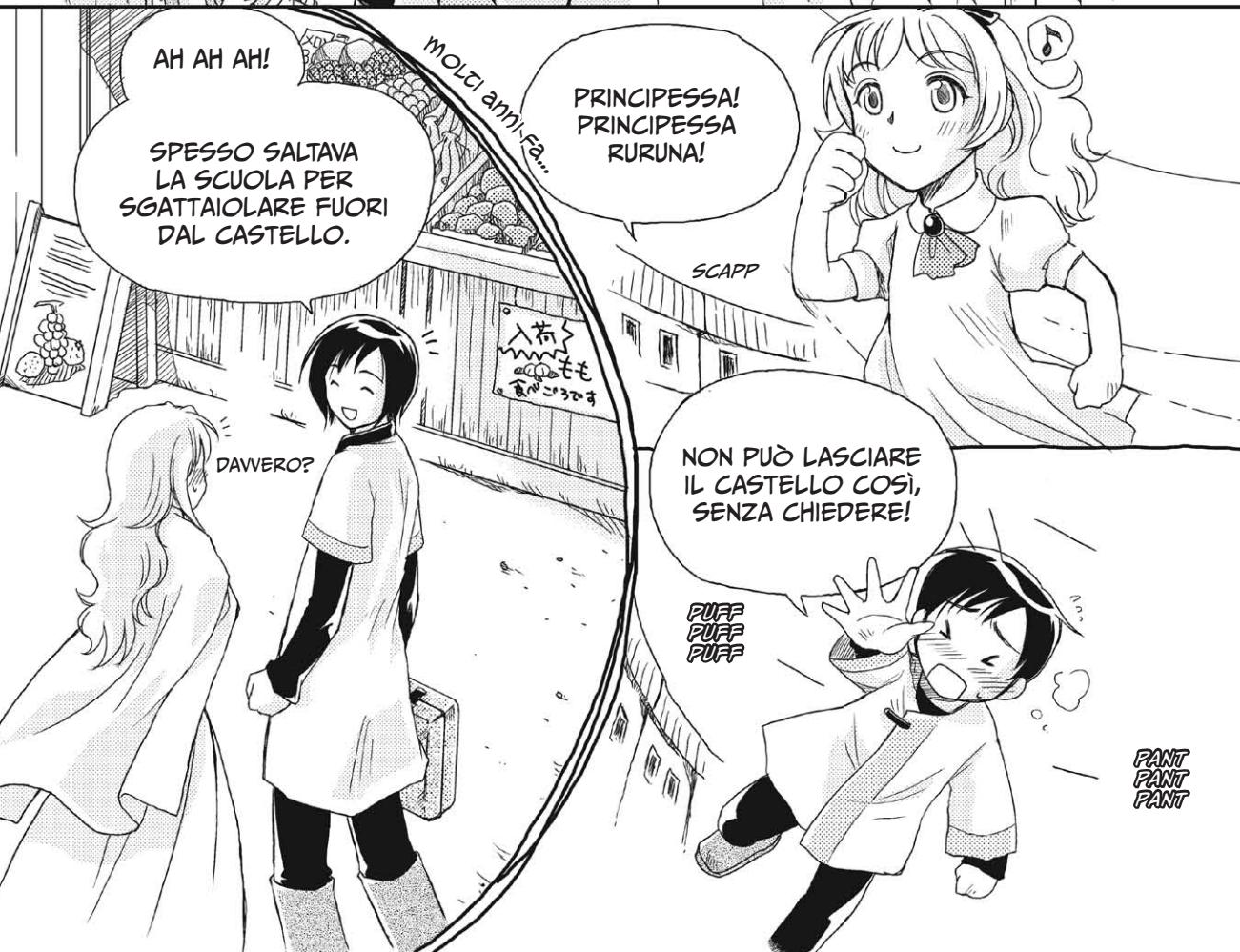
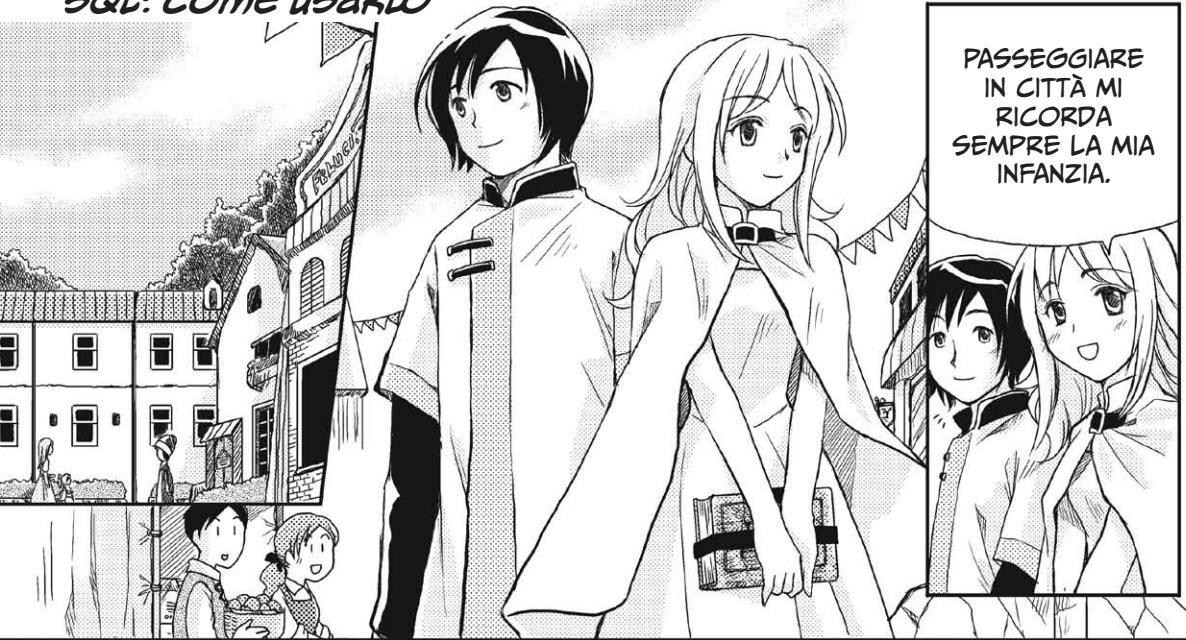
In questa fase ti viene chiesto di svolgere diversi altri compiti, oltre alla progettazione delle tabelle. Per esempio, devi considerare il tipo di dato da usare nella tabella. Dovrai anche specificare quali colonne contengono valori numerici, quali valute o campi di testo. Dovrai anche ideare un metodo che ti consenta di eseguire ricerche in modo veloce. A volte dovrà progettare il tuo database tenendo in considerazione l'organizzazione fisica dei file. Dovrai inoltre controllare l'accesso ai dati dei vari utenti, per ragioni di sicurezza. Ci sono molti fattori da tenere in considerazione quando si progetta un database. Nei capitoli seguenti ne affronteremo alcuni.

4

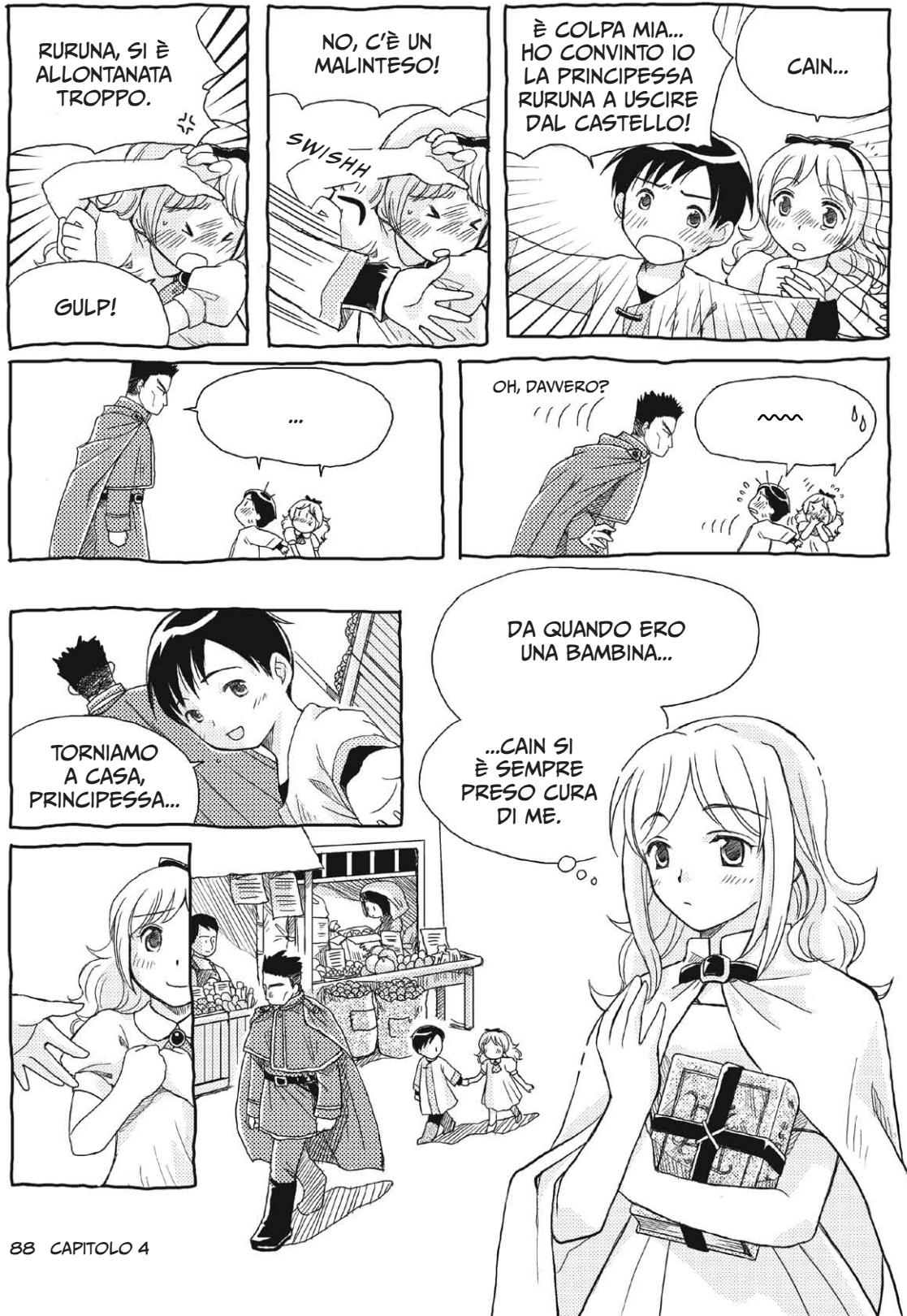
IMPARIAMO IL LINGUAGGIO SQL!

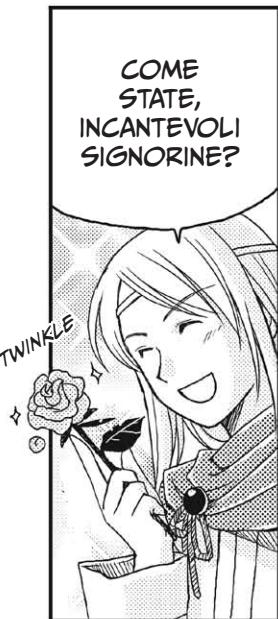
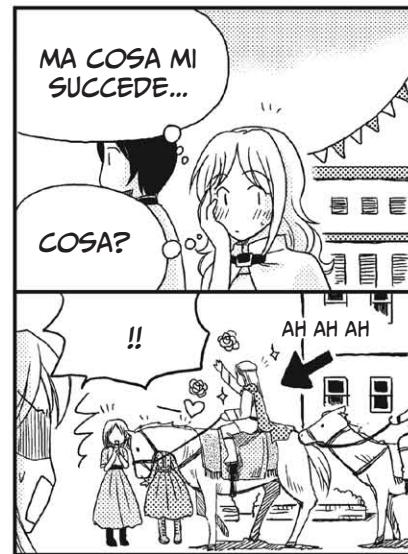


SQL: COME USARLO

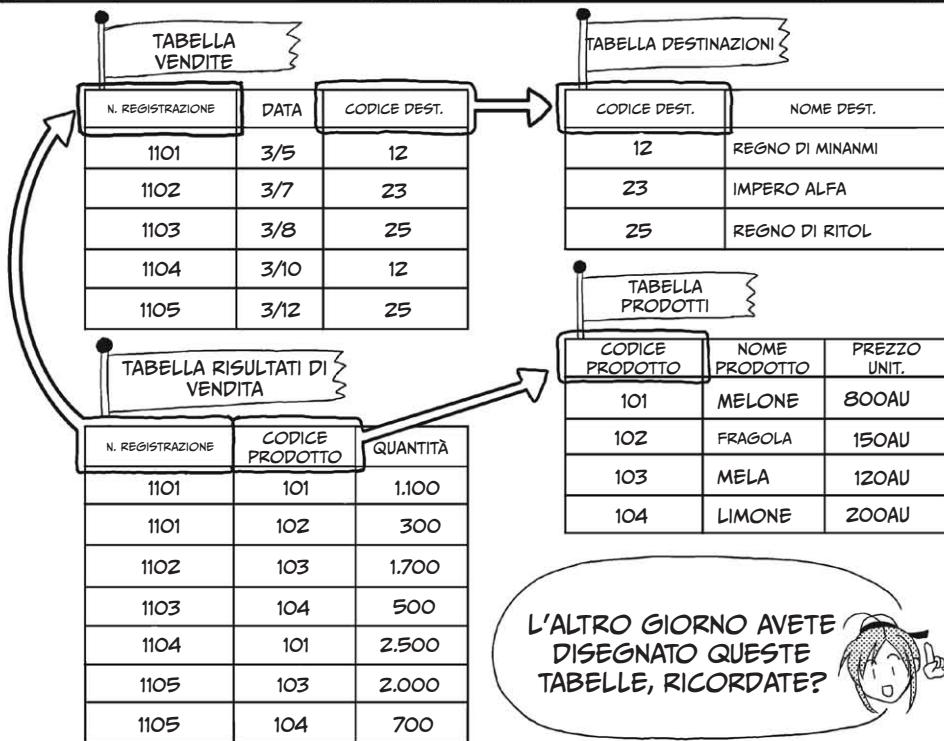
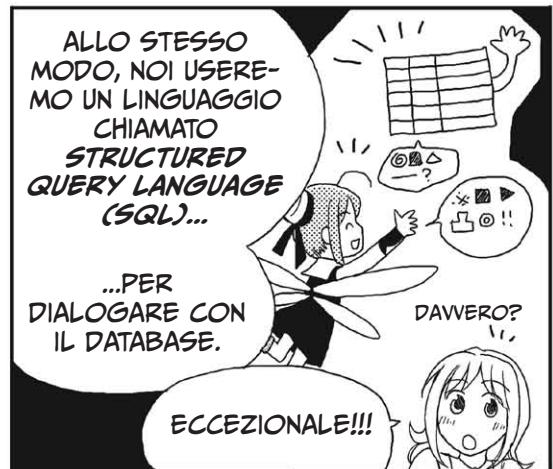














MA ORA AVETE BISOGNO
DI SQL PER INSERIRE
QUESTE TABELLE E I DATI
CHE CONTENGONO NEL
DATABASE.

SQL PERMETTE DI

- CREARE TABELLE
- INSERIRE ED ESTRARRE DATI
- GESTIRE GLI UTENTI



USANDO SQL POTRETE
DIALOGARE CON
IL DATABASE, IN
RELAZIONE A QUESTI
TRE COMPITI...

SEMPRA PROPRIO
CHE SI POSSA FARE
DI TUTTO!



MA... SI
DIREBBE UN
SACCO DI
LAVORO.

NON C'È
PROBLEMA.



ABBIAMO
IMPARATO UN
SACCO DI COSE!



BE'... È VERO...
E VOGLIO
COMINCIARE PRIMA
POSSIBILE A USARE
IL MIO DATABASE.

ESATTO.

QUESTO
È LO
SPIRITO
GIUSTO!



HO GIÀ INSERITO LE
TABELLE E I DATI CHE
ABBIAMO REDATTO.

BIP
VEDIAMO COME
RECUPERARLI.

LA RICERCA DEI DATI

DOBBIAMO RECUPERARE SOLO I NOMI DEI PRODOTTI PER CREARE UNA LISTA DEI NOMI PRODOTTO USANDO SQL.

COME SI FA?

BASTA CHIEDERE AL DATABASE DI ESTRARRE LA COLONNA CON I NOMI DEI PRODOTTI...

PER FAVORE...

...DALLA TABELLA PRODOTTI.

SIGNOR DATABASE...

...TI PREGHIAMO DI ESTRARRE LA COLONNA COI NOMI DEI PRODOTTI...

MA NON DOVETE PREGARE! BASTA USARE SQL...

BISOGNA SCRIVERE COSÌ:

```
SELECT nome_prodotto  
FROM prodotti;
```

IN SQL, OGNI RICHIESTA CHE VIENE FATTA SI CHIAMA ISTRUZIONE (O COMANDO).

QUEST'ISTRUZIONE SQL CONSISTE DI DUE GRUPPI DI PAROLE: **SELECT NOME_PRODOTTO E FROM PRODOTTI**.

QUESTI GRUPPI DI PAROLE SONO CHIAMATI **FRASI**.

IN SQL DEVI SPECIFICARE IL NOME DELLA COLONNA CHE VUOI ESTRARRE CON LA FRASE **FROM**.

FROM

TABELLA PRODOTTI

CODICE PRODOTTO	NOME PRODOTTO	PREZZO UNIT.
101	MELONE	800AU
102	FRAGOLA	150AU
103	MELA	120AU
104	LIMONE	200AU

SELECT

ED ECCO I DATI CHE OTTENIAMO.

SIAMO RIUSCITI A RECUPERARE TUTTI I NOMI DALLA TABELLA PRODOTTI.

ECCO FATTO!

NOME PRODOTTO

MELONE

FRAGOLA

MELA

LIMONE

STIAMO DIALOGANDO CON IL DATABASE USANDO SQL.

PROPRIO COSÌ, ESISTONO MOLTI TIPI DI FRASE PER RECUPERARE I DATI NECESSARI.

VARI TIPI... MMM.

ALLORA, PER ESEMPIO,

SE DOVESSIMO CHIEDERE UNA LISTA DEI PRODOTTI CHE HANNO UN PREZZO UNITARIO MAGGIORALE O UGUALE A 200AU?

MAGGIORALE O UGUALE A 200AU

....
....
....
....
....
....
....
....
....
....



IN QUESTO CASO
NON TI INTERESSANO
I DATI DI TUTTI I
PRODOTTI.

VUOI SOLO
QUELLI CHE
HANNO UN
PREZZO UNITARIO
UGUALE O
SUPERIORE A
ZOOAU.

SÌ, ESATTO.

IN QUESTI CASI DEVI
SPECIFICARE LE TUE
CONDIZIONI USANDO
LA FRASE WHERE.
PER ESEMPIO:

WHERE

WHERE prezzo_unitario>=200

SI SCRIVE COSÌ.

CAPISCO,
MA...

NON È UN PO'
SCOMODO DOVER
SPECIFICARE IL NOME
DELLA COLONNA TUTTE
LE VOLTE?

NESSUN
PROBLEMA! PER
SELEZIONARE
TUTTE LE
COLONNE,

MMM...

È UNA
SCOCCIATURA!

BASTA USARE IL
CARATTERE "*"! ECCO
COME SI FA.

| X!!
BANG!!

```
SELECT *  
FROM prodotti  
WHERE prezzo_unitario>=200
```

QUINDI,

GRAZIE A QUESTA
ISTRUZIONE
RECUPERERAI DALLA
TABELLA TUTTI I
PRODOTTI...

ECCO QUI

PRODOTTI CHE COSTANO ZOOAU
O PIÙ

CODICE PRODOTTO	NOME PRODOTTO	PREZZO UNIT.
101	MELONE	800AU
104	LIMONE	ZOOAU

...CHE HANNO UN
PREZZO UNITARIO
UGUALE O SUPERIORE
A ZOOAU.

QUINDI ALLO
STESO MODO
PUOI ESTRARRE
I PRODOTTI CHE
COSTANO MENO DI
200AU.

WHERE prezzo_unitario<200

ESATTO, BASTA
FARE COSÌ!

DOBBIAMO IMPARARE
A SCRIVERE LE
CONDIZIONI.

INFATTI...

...ALLORA COME
FACCIO PER ESEMPIO
A ESTRARRE MELA?

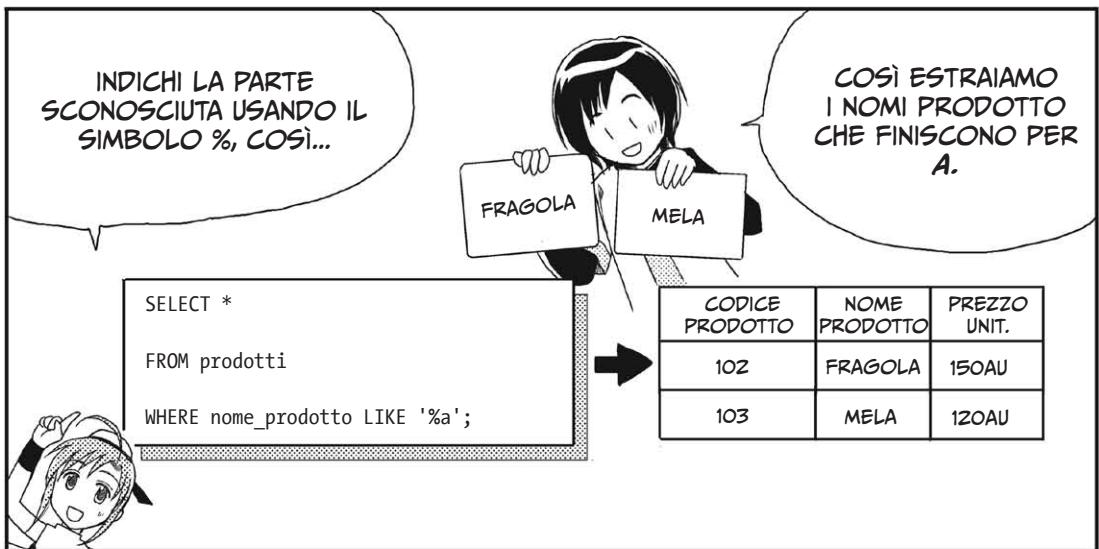
```
SELECT *  
FROM prodotti  
WHERE nome_prodotto='mela';
```

DEVI SCRIVERE COSÌ.
QUANDO USI I CARATTERI
NELLE CONDIZIONI DEVI
SEMPRE METTERLI TRA
DUE APICI ('').

CODICE PRODOTTO	NOME PRODOTTO	PREZZO UNIT.
103	MELA	120AU

IN QUESTO MODO
RECUPERIAMO LE
MELE!

ESATTAMENTE.



L'ESTRAZIONE DEI DATI

POTETE ANCHE ORDINARE I RISULTATI GRAZIE ALL'ISTRUZIONE *ORDER BY*.

PER ESTRARRE I PRODOTTI ORDINATI PER PREZZO CRESCENTE, BASTA AGGIUNGERE L'ISTRUZIONE *ORDER BY PREZZO UNITARIO*.

SI POSSONO RECUPERARE LE INFORMAZIONI SUI PRODOTTI IN QUESTO MODO.



```
SELECT *  
FROM prodotti  
WHERE nome_prodotto LIKE '%a';  
ORDER BY prezzo_unitario;
```

OTTIMO!

CODICE PRODOTTO	NOME PRODOTTO	PREZZO UNIT.
103	MELA	120AU
102	FRAGOLA	150AU

VOGLIO SAPERE DI PIÙ SU SQL, TICO!

VERAMENTE?

MI FA PIACERE!

CHE NE DICI DI QUESTO?

SE INSIEME A SELECT USI AVG(NOME COLONNA) PUOI OTTENERE LA MEDIA DEI VALORI DI TUTTE LE RIGHE.

```
SELECT AVG(prezzo_unitario)  
FROM prodotti;
```

ば"ん!!

È INCREDIBILE.

ZUUPP!!

PREZZO UNITARIO MEDIO

317,5

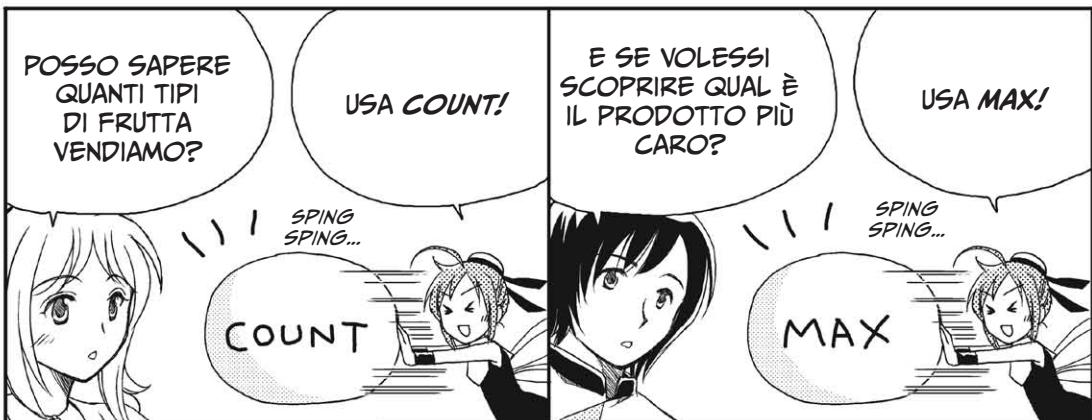
ECCO IL PREZZO UNITARIO MEDIO DEI PRODOTTI.



FUNZIONI DI AGGREGAZIONE IN SQL

Funzione	Descrizione
COUNT(*)	Restituisce il numero di righe
COUNT(nome_colonna)	Restituisce il numero di volte in cui la colonna contiene un valore
COUNT(DISTINCT nome_colonna)	Restituisce il numero di valori distinti nella colonna
SUM(nome_colonna)	Restituisce la somma dei valori di tutte le righe nella colonna
AVG(nome_colonna)	Restituisce la media dei valori di tutte le righe nella colonna
MAX(nome_colonna)	Restituisce il valore massimo nella colonna
MIN(nome_colonna)	Restituisce il valore minimo nella colonna





UNIRE LE TABELLE

PER CREARE UN RESOCONTO DELLE VENDITE BISOGNA RECUPERARE I DATI UNENDO LA TABELLA PRODOTTI ALLE ALTRE, CIOÈ LA TABELLA DESTINAZIONI, LA TABELLA VENDITE E LA TABELLA RISULTATI DI VENDITA.

UH-OH!

GIUSTO, PRIMA DELLA NORMALIZZAZIONE AVEVAMO UNA SOLA TABELLA.

PER UNIRE LE TABELLE, SQL RICHIEDE UN PREREQUISITO...

LA CHIAVE PRIMARIA DEV'ESSERE UGUALE ALLA CHIAVE ESTERNA CHE RIFERISCE ALLA CHIAVE PRIMARIA.

COME SI FA A SPECIFICARLO?

UNISCI LE TABELLE ELENCONDOLE SEPARATE DA UNA VIRGOLA.

SE LA STESSA COLONNA APPARE IN PIÙ DI UNA TABELLA, LO SPECIFICHI COME NOME_TABELLA.
NOME_COLONNA.

```
SELECT vendite.numero_registrazione, data, vendite.codice_destinatario,  
nome_destinatario, risultati_vendita.codice_prodotto,  
nome_prodotto, prezzo_unitario  
  
FROM vendite, risultati_vendita, prodotti, destinazioni  
  
WHERE vendite.numero_registrazione = risultati_vendita.numero_registrazione  
AND  
risultati_vendita.codice_prodotto = prodotti.codice_prodotto  
AND  
destinazioni.codice_destinatario = vendite.codice_destinatario
```

UNENDO LE QUATTRO TABELLE POI POTREMO RESTRINGERE IL RISULTATO USANDO WHERE.



IN QUESTO MODO POTETE
RECUPERARE I RISULTATI DI
VENDITA DALLE TABELLE ANCHE
QUANDO SONO DIVISE.

N. REGISTRAZIONE	DATA	CODICE DEST.	NOME DEST.	CODICE PRODOTTO	NOME PRODOTTO	PREZZO UNIT.	QUANTITÀ
1101	3/5	12	REGNO DI MINANMI	101	MELONE	800AU	1.100
1101	3/5	12	REGNO DI MINANMI	102	FRAGOLA	150AU	300
1102	3/7	23	IMPERO ALFA	103	MELA	120AU	1.700
1103	3/8	25	REGNO DI RITOL	104	LIMONE	200AU	500
1104	3/10	12	REGNO DI MINANMI	101	MELONE	800AU	2.500
1105	3/12	25	REGNO DI RITOL	103	MELA	120AU	2.000
1105	3/12	25	REGNO DI RITOL	104	LIMONE	200AU	700

È LA STESSA TABELLA
DALLA QUALE SIAMO
PARTITI, L'ABBIAMO
RICREATA!

POSSIAMO RECUPERARE
I DATI RELATIVI ALLE VENDITE
ANCHE GESTENDO I PRODOTTI,
LE DESTINAZIONI E LE VENDITE IN
MODO AUTONOMO E
INDIPENDENTE.

SPLENDIDO!

WOW!

CREARE UNA TABELLA



```
CREATE TABLE prodotti
(
codice_prodotto int NOT NULL,
nome_prodotto varchar(255),
prezzo_unitario int,
PRIMARY KEY(codice_prodotto)
);
```

CODICE PRODOTTO	NOME PRODOTTO	PREZZO UNIT.

DEVI SPECIFICARE ANCHE
QUAL È LA CHIAVE PRIMARIA,
IN QUESTO CASO È IL
CODICE PRODOTTO.*

ABBIAMO ANCHE DEFINITO I
TIPI DI DATO CONTENUTI IN
CIASCUNA COLONNA. POTETE
VEDERE CHE IL CODICE
PRODOTTO E IL PREZZO
SONO NUMERI INTERI (INT).
VARCHAR INDICA UN CAMPO DI
TESTO, LIMITATO A MASSIMO
255 CARATTERI (255).

COSÌ...

PER IMPEDIRE DI
INSERIRE VALORI
NON CORRETTI.

* VEDI PAGINA 115 PER UNA SPIEGAZIONE DETTAGLIATA DELLE ISTRUZIONI PER CREARE LE TABELLE.

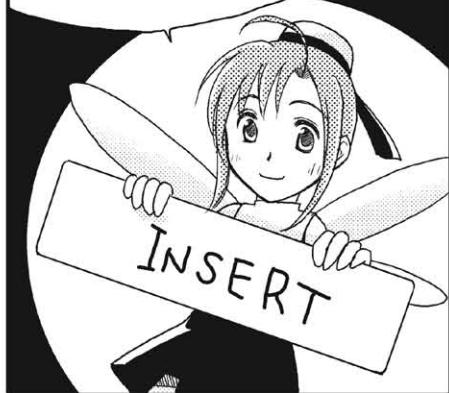
ORA POSSIAMO
INSERIRE I DATI NELLA
TABELLA, GIUSTO?



PER FARLO, USEREMO
L'ISTRUZIONE **INSERT**.



CODICE PRODOTTO	NOME PRODOTTO	PREZZO UNITARIO
PROPRIO COSÌ.		



```
INSERT INTO prodotti (codice_prodotto, nome_prodotto,  
                     prezzo_unitario)  
VALUES (101, 'melone', 800);
```

PUOI ANCHE
CANCELLARE (CON
L'ISTRUZIONE **DELETE**) E
AGGIORNARE (ISTRUZIONE
UPDATE) I DATI.



CODICE PRODOTTO	NOME PRODOTTO	PREZZO UNITARIO
101	MELONE	800AU



MELONE È STATO
INSERITO NELLA
TABELLA PRODOTTI
COSÌ.



E USANDO SQL
PUOI MODIFICARE IL
PREZZO UNITARIO.



PANORAMICA SU SQL



In questo capitolo la Principessa Ruruna e Cain hanno imparato a usare *SQL (Structured Query Language)*, un linguaggio usato per operare sui database relazionali. I comandi SQL si possono suddividere in tre gruppi:

Data Definition Language (DDL) per creare una tabella

Data Manipulation Language (DML) per inserire ed estrarre i dati

Data Control Language (DCL) per gestire l'accesso da parte degli utenti

SQL possiede le istruzioni per creare la struttura di un database e un'istruzione specifica per creare tabelle all'interno di un database. Puoi anche usare questo linguaggio per modificare o cancellare una tabella. Il linguaggio deputato a queste funzioni si chiama *Data Definition Language (DDL)*.

SQL include anche istruzioni per inserire, modificare e cancellare i dati, nonché un comando che consente la ricerca dei dati. Il linguaggio deputato a queste funzioni si chiama *Data Manipulation Language (DML)*.

Per finire, SQL offre la possibilità di controllare un database in modo da scongiurare conflitti tra i dati anche quando più persone lo stanno utilizzando contemporaneamente. Il linguaggio deputato a queste funzioni si chiama *Data Control Language (DCL)*.

LA RICERCA DEI DATI CON L'ISTRUZIONE SELECT

La Principessa Ruruna e Cain hanno cominciato a impraticarsi con SQL usando una funzione elementare di ricerca dati. Per effettuare una ricerca con SQL bisogna inserire una cosiddetta *istruzione* (o combinazione di frasi). Per ricercare un prodotto specifico con un prezzo unitario di 200Au, per esempio, bisogna usare la seguente istruzione SQL.

```
SELECT *
FROM prodotti
WHERE prezzo_unitario=200
```

Un'istruzione SQL è una combinazione di frasi.

L'istruzione SELECT è la più elementare delle istruzioni SQL. Specifica *quale colonna, da quale tabella (FROM)* e sulla base di *quali condizioni (WHERE)*. Puoi combinare queste frasi per interrogare un database in modo molto intuitivo (anche un utente non molto avvezzo all'uso dei database può utilizzare quest'istruzione per recuperare dati).

STABILIRE LE CONDIZIONI

Come ha detto Cain, "dobbiamo imparare a scrivere le condizioni". Scopriamo alcuni modi per farlo in SQL.

GLI OPERATORI DI CONFRONTO

Un modo per esprimere le condizioni è usare *operatori di confronto* come \geq e $=$. Per esempio, la condizione "A è maggiore o uguale a B" si può esprimere usando \geq , mentre la condizione "A è uguale a B" si può esprimere usando $=$. Troverai altri esempi di operatori di confronto nella seguente tabella.

OPERATORI DI CONFRONTO

Operatore di confronto	Descrizione	Esempio	Spiegazione dell'esempio
A = B	A è uguale a B	prezzo_unitario=200	Il prezzo unitario è uguale a 200Au.
A > B	A è maggiore di B	prezzo_unitario>200	Il prezzo unitario è maggiore di 200Au.
A \geq B	A è maggiore o uguale a B	prezzo_unitario \geq 200	Il prezzo unitario è maggiore o uguale a 200Au.
A < B	A è minore di B	prezzo_unitario<200	Il prezzo unitario è minore di 200Au.
A \leq B	A è minore o uguale a B	prezzo_unitario \leq 200	Il prezzo unitario è minore o uguale a 200Au.
A \neq B	A è diverso da B	prezzo_unitario \neq 200	Il prezzo unitario è diverso da 200Au.

OPERATORI LOGICI

In certi casi è necessario esprimere condizioni più complesse del semplice confronto. Puoi usare gli *operatori logici* (*AND*, *OR* e *NOT*) per creare condizioni combinate più elaborate, come evidenziato nella seguente tabella.

OPERATORI LOGICI

Operatore logico	Descrizione	Esempio	Spiegazione dell'esempio
AND	A e B	Codice prodotto \geq 200 AND prezzo unitario = 100	Il codice prodotto è maggiore o uguale a 200 e il prezzo unitario è 100Au.
OR	A o B	Codice prodotto \geq 200 OR prezzo unitario = 100	Il codice prodotto è maggiore o uguale a 200 o il prezzo unitario è 100Au.
NOT	Non A	NOT prezzo unitario = 100	Il prezzo unitario non è 100Au.

I PATTERN

Quando non sai esattamente cosa vuoi cercare, puoi anche usare sequenze particolari di caratteri jolly, dette pattern. Quando esegui la ricerca con i pattern avrai a che fare con caratteri come % oppure _ associati all'istruzione LIKE. Questo ti permetterà di cercare una stringa di caratteri che corrisponda al pattern specificato. Per esempio, puoi cercare un valore che corrisponde a una stringa, specificata anche solo parzialmente, usando %, che indica una stringa di caratteri di una lunghezza qualsiasi, e _, che specifica invece un solo carattere.

Qui sotto trovi un esempio dell'uso di questi caratteri jolly. L'istruzione ricerca una stringa di caratteri che definisce un nome prodotto che finisce per e.

SELECT *
FROM prodotti
WHERE nome_prodotto LIKE '%e';

La regola è definita grazie a un carattere jolly.

Codice prodotto	Nome prodotto	Prezzo unit.
101	Melone	800Au
104	Limone	200Au

I caratteri jolly disponibili in SQL sono spiegati qui sotto.

CARATTERI JOLLY

Carattere jolly	Descrizione	Esempio di pattern	Stringa valida
%	Corrisponde a una stringa di lunghezza qualsiasi	%e m%	Limone Melone Mela Melone
_	Corrisponde a un carattere	_o o_	io on

LE RICERCHE

Esistono molti altri metodi per ricercare dati. Per esempio, puoi specificare un intervallo di valori con BETWEEN X AND Y. Se specifichi l'intervallo come mostrato nell'esempio seguente, puoi estrarre i prodotti con prezzo unitario maggiore o uguale a 150Au e minore di 200Au.

SELECT *
FROM prodotti
WHERE prezzo_unitario
BETWEEN 150 AND 200;

Specifica un intervallo di ricerca.

Puoi anche specificare che stai cercando i prodotti il cui prezzo unitario è zero, grazie all'operatore NULL.

SELECT *
FROM prodotti
WHERE prezzo_unitario is NULL;

Cerca un valore vuoto.



DOMANDE

Adesso prova a creare delle istruzioni SQL usando diversi tipi di condizioni. Partiamo dalla Tabella Destinazioni riportata qui sotto (l'unità di misura per la popolazione è 10.000 abitanti). Rispondi alle domande usando istruzioni SQL. Troverai le risposte a pagina 119.

TABELLA DESTINAZIONI

Codice destinatario	Nome destinatario	Popolazione
12	Regno di Minanmi	100
23	Impero Alfa	120
25	Regno di Ritol	150
30	Regno di Sazanna	80

D1

Estrai la tabella qui sotto per trovare gli stati con un numero di abitanti maggiore o uguale a 1 milione.

Codice destinatario	Nome destinatario	Popolazione
12	Regno di Minanmi	100
23	Impero Alfa	120
25	Regno di Ritol	150

D2

Estrai la tabella qui sotto per trovare gli stati con un numero di abitanti minore di 1 milione.

Codice destinatario	Nome destinatario	Popolazione
30	Regno di Sazanna	80

D3

Trova gli stati in cui il codice destinatario è minore di 20 e il numero di abitanti maggiore o uguale a 1 milione.

D4

Trova gli stati in cui il codice destinatario è maggiore o uguale a 30 e il numero di abitanti maggiore di 1 milione.

D5

Quanti sono gli abitanti del Regno di Ritol?

D6

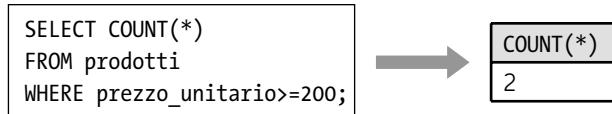
Trova gli stati i cui nomi contengono la lettera *n*.

LE FUNZIONI DI AGGREGAZIONE



La Principessa Ruruna e Cain hanno scoperto diverse funzioni di aggregazione. Tali funzioni sono anche chiamate *funzioni di colonna*, e possono essere usate per aggregare informazioni come il valore massimo, il valore minimo, il numero di oggetti e la somma.

Utilizzando WHERE in associazione a una funzione aggregata potrai ottenere un valore aggregato soltanto per le righe desiderate. Nell'esempio qui sotto puoi scoprire quanti sono i prodotti il cui prezzo è maggiore o uguale a 200Au.



AGGREGARE I DATI CON L'OPERATORE DI RAGGRUPPAMENTO

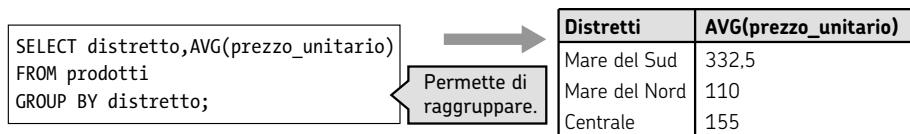
Raggruppando i dati potrai facilmente ottenere valori aggregati. Se per esempio vuoi estrarre il numero di prodotti e il loro prezzo unitario medio a seconda del distretto, potrai utilizzare la funzione di raggruppamento.

Per raggruppare i dati è possibile combinare la funzione di aggregazione con la frase GROUP BY. Partiamo dalla tabella qui sotto.

TABELLA PRODOTTI

Codice prodotto	Nome prodotto	Prezzo unit.	Distretto
101	Melone	800Au	Mare del Sud
102	Fragola	150Au	Centrale
103	Mela	120Au	Mare del Nord
104	Limone	200Au	Mare del Sud
201	Castagna	100Au	Mare del Nord
202	Caco	160Au	Centrale
301	Pesca	130Au	Mare del Sud
302	Kiwi	200Au	Mare del Sud

Per ottenere il prezzo unitario medio per ognuno dei distretti della Tabella Prodotti, specifica la colonna Distretti e usa la funzione AVG comandata dalla frase GROUP BY. In questo modo i dati saranno raggruppati per distretto e otterrai il valore medio relativo a ogni singolo distretto.



E se volessi ulteriormente restringere questi risultati sulla base di una proprietà particolare? Immagina di voler trovare i prodotti il cui prezzo unitario medio regionale è maggiore o uguale a 200Au. In questo caso non dovrà specificare una condizione nella frase WHERE, ma usare al suo posto la frase HAVING. Questo ti permetterà di estrarre solo i distretti in cui la media rispetta il tuo requisito.

```
SELECT distretto, AVG(prezzo_unitario)
FROM prodotti
GROUP BY distretto;
HAVING AVG(prezzo_unitario)>=200;
```



Filtra i risultati già raggruppati.

Distretto	AVG(prezzo_unitario)
Mare del Sud	332,5

DOMANDE



Rispondi alle seguenti domande usando questa Tabella Destinazioni Esportazione (l'unità di misura per la popolazione è 10.000 abitanti). Troverai le risposte a pagina 120.

TABELLA DESTINAZIONI ESPORTAZIONE

Codice destinatario	Nome destinatario	Popolazione	Distretto
12	Regno di Minanmi	100	Mare del Sud
15	Regno di Paronu	200	Centrale
22	Regno di Tokanta	160	Mare del Nord
23	Impero Alfa	120	Mare del Nord
25	Regno di Ritol	150	Mare del Sud
30	Regno di Sazanna	80	Mare del Sud
31	Regno di Taharu	240	Mare del Nord
33	Regno di Mariyon	300	Centrale

D7

Quanti abitanti ha lo stato meno popoloso?

D8

Quanti abitanti ha lo stato più popoloso?

D9

Qual è il totale degli abitanti di tutti gli stati presenti nella Tabella Destinazioni Esportazione?

D10

Qual è la popolazione totale degli stati il cui codice destinatario è maggiore di 20?

D11

Quanti stati presentano una popolazione maggiore o uguale a 1 milione?

D12

Quanti stati ci sono nel distretto del Mare del Nord?

D13

Qual è lo stato più popoloso del distretto del Mare del Nord?

D14

Qual è la popolazione totale di tutti gli stati escluso il Regno di Ritol?

D15

Trova i distretti in cui la popolazione media è maggiore o uguale a 2 milioni.

D16

Trova i distretti che contengono almeno tre stati.

La ricerca dei dati



In aggiunta a quelli che abbiamo già visto, esistono in SQL altri metodi più sofisticati per interrogare un database.

USARE UNA SUBQUERY

Per esempio, è possibile inserire un'interrogazione (query) all'interno di un'altra interrogazione, ottenendo così una *subquery*. Diamo un'occhiata alle tabelle seguenti.

TABELLA PRODOTTI

Codice prodotto	Nome prodotto	Prezzo unit.
101	Melone	800Au
102	Fragola	150Au
103	Mela	120Au
104	Limone	200Au

TABELLA RISULTATI DI VENDITA

N. Registrazione	Codice prodotto	Quantità
1101	101	1.100
1101	102	300
1102	103	1.700
1103	104	500
1104	101	2.500
1105	103	2.000
1105	104	700

Puoi usare queste due tabelle per recuperare i nomi dei prodotti le cui vendite superano o sono uguali a 1.000 unità. La seguente istruzione SQL eseguirà la ricerca.

```
SELECT * FROM prodotti
WHERE codice_prodotto IN
(SELECT codice_prodotto
FROM risultati_vendita
WHERE quantita>=1000);
```

Quest'istruzione contiene una subquery.

In quest'istruzione SQL viene prima eseguita l'espressione SELECT tra parentesi. Il codice prodotto nella Tabella Risultati di Vendita viene cercato per primo, e da questa ricerca vengono estratti i codici 101 e 103 (sono gli unici in cui il venduto supera le 1.000 unità).

Questi codici prodotto vengono poi usati come parte della condizione dell'istruzione SELECT fuori dalle parentesi. IN indica che la condizione è soddisfatta quando un record corrisponde a uno dei valori inclusi tra parentesi. Quindi i prodotti corrispondenti ai codici prodotto 101 e 103 saranno restituiti come risultato della query intera.

In altre parole, quando c'è una subquery il risultato dell'istruzione SELECT tra parentesi verrà trasmesso all'altra istruzione SELECT per continuare la ricerca. Il risultato di questa ricerca è esposto nella tabella seguente.

Codice prodotto	Nome prodotto	Prezzo unit.
101	Melone	800Au
103	Mela	120Au

USARE UNA SUBQUERY CORRELATA

Consideriamo adesso una subquery *contenuta all'interno* di un'altra query. Questa potrebbe far riferimento a dati ottenuti da un'altra query, è il caso delle *subquery correlate*. Nella query seguente, la tabella risultati_vendita nella query esterna viene temporaneamente ribattezzata U per permettere alla subquery di riferirsi a lei in modo non ambiguo. La sintassi U.codice_prodotto indica a quale colonna codice_prodotto ci stiamo riferendo, visto che esistono due sorgenti per quella colonna all'interno della subquery.

Visto che la subquery si riferisce a dati provenienti dall'altra query, la subquery non è indipendente dalla query esterna, come negli esempi precedenti. Questa dipendenza è chiamata *correlazione*.

```

❶ SELECT *
FROM risultati_vendita U

❷ WHERE quantita>

❸ (SELECT AVG(quantita)
FROM risultati_vendita
WHERE codice_prodotto=U.codice_prodotto);

```

# registrazione	Codice prodotto	Quantità
1104	101	2.500
1105	103	2.000
1105	104	700

Questa query restituisce le registrazioni in cui il volume di vendita è superiore alla media per quel prodotto.

Vediamo come viene elaborata questa subquery. In una subquery correlata viene prima eseguita la query esterna.

```

❶   SELECT *
FROM risultati_vendita U

```

Il risultato è mandato alla query interna per essere valutato riga per riga. Analizziamo ora come viene elaborata la prima riga, ovvero il codice prodotto 101.

③ (SELECT AVG(quantita)
FROM risultati_vendita
WHERE codice_prodotto=101)

Il codice prodotto nella prima riga è 101 (melone). Il venduto medio per i meloni è pari a 1.800 unità e questo risultato è ora inviato come condizione alla query esterna.

② WHERE quantita>(1.800)

Questo processo viene iterato per tutte le righe della Tabella Risultati di Vendita: i passi **②** e **③** vengono dunque eseguiti per tutti i codici prodotto. In altri termini, questa query estrae le registrazioni relative a frutti che sono stati venduti in misura maggiore alla media di vendita per quel frutto particolare.

Di conseguenza solo la quinta, la sesta e la settima colonna di **①** sono estratte.



DOMANDE

Adesso rispondi alle seguenti domande, basate sulla Tabella Prodotti e sulla Tabella Risultati di Vendita. Troverai le risposte a pagina 122.

D17

Trova i risultati di vendita dei frutti il cui prezzo unitario è maggiore o uguale a 300Au ed estrai la tabella qui sotto.

Registrazione	Codice prodotto	Quantità
1101	101	1.100
1104	101	2.500

D18

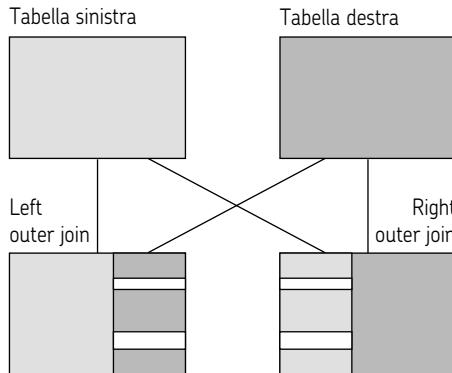
Ottieni i volumi medi di vendita suddivisi per prodotto, e trova i prodotti che presentano volumi di vendita inferiori alla media.

UNIRE LE TABELLE

Dopo aver condotto una ricerca grazie a SQL, la Principessa Ruruna e Cain hanno generato un resoconto delle vendite combinando le tabelle. Unire le tabelle combinando colonne con lo stesso nome è un'operazione chiamata *equi-join*. In un'equi-join la condizione dev'essere l'uguaglianza delle righe. Riunire in un'unica colonna più colonne dallo stesso nome è invece definita *natural join*.

Il metodo con cui vengono selezionate solo righe aventi un valore in comune (come nel caso di una equi join) si chiama *inner join*.

Invece, il metodo che mantiene tutte le righe di una tabella e specifica un valore vuoto per le righe non incluse in un'altra tabella si chiama *outer join*. Se metti una tabella creata grazie a un outer join a destra o a sinistra di un'istruzione SQL, otterrai una *left outer join* o una *right outer join*, a seconda di quali righe vengono mantenute.



CREARE UNA TABELLA



Alla fine la Principessa Ruruna e Cain hanno imparato a usare l'istruzione che consente di creare una tabella, CREATE TABLE. La sintassi di un'istruzione CREATE TABLE spesso dipende dal tipo di database in uso. Vediamo un esempio:

```
CREATE TABLE prodotti
(
codice_prodotto int NOT NULL,
nome_prodotto varchar(255),
prezzo_unitario int,
PRIMARY KEY(codice_prodotto)
);
```

Quest'istruzione crea una tabella.

Quando crei una tabella devi sempre specificare i nomi delle colonne. Puoi anche specificare una chiave primaria e una chiave esterna per ogni colonna. In questo esempio il codice prodotto viene specificato come chiave primaria e il valore del codice prodotto non può essere null. Quando crei una tabella spesso hai bisogno di includere una delle seguenti condizioni.

VINCOLI SUI DATI DI UNA TABELLA

Vincolo	Descrizione
PRIMARY KEY	Stabilisce una chiave primaria
UNIQUE	Deve essere univoco
NOT NULL	Non accetta un valore NULL
CHECK	Verifica che il valore stia in un intervallo
DEFAULT	Inserisce un valore di default predefinito
FOREIGN KEY REFERENCES	Stabilisce una chiave esterna

Queste condizioni sono chiamate *vincoli*. Stabilire dei vincoli quando si crea una tabella consente di evitare conflitti sui dati e aiuta nella corretta gestione del database.

INSERIRE, AGGIORNARE O CANCELLARE LE RIGHE

Puoi usare le istruzioni INSERT, UPDATE e DELETE per inserire, aggiornare o cancellare dati da una tabella creata usando l'istruzione CREATE TABLE. Vediamo come inserire, aggiornare e cancellare i dati usando SQL.

```
INSERT INTO prodotti  
(codice_prodotto, nome_prodotto, prezzo_unitario)  
VALUES (200, 'cileggia', 200);
```

Quest'istruzione aggiunge 'cileggia'.

```
UPDATE prodotti  
SET nome_prodotto='cantalupo'  
WHERE nome_prodotto='melone';
```

Quest'istruzione modifica 'melone' in 'cantalupo'.

```
DELETE FROM prodotti  
WHERE nome_prodotto='mela';
```

Quest'istruzione cancella 'mela'.

Codice prodotto	Nome prodotto	Prezzo unit.
101	Cantalupo	800G
102	Fragola	150G
103	Mela	120G
104	Limone	200G
200	Cileggia	200G

Modificato in cantalupo.

Mela cancellata.

Cileggia aggiunta.

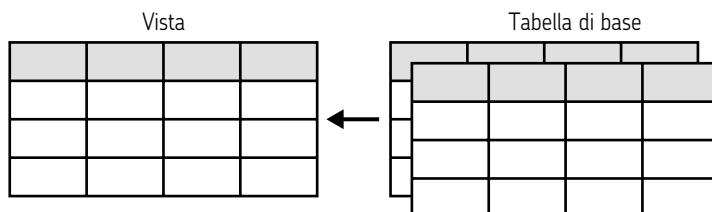
Quando inserisci, aggiorni o cancelli una riga, non puoi violare i vincoli espressi dalla istruzione CREATE TABLE. Se un prodotto con codice prodotto 200 esiste già, non riuscirai ad aggiungere ciliegia, visto che non è possibile avere duplicati fra le chiavi primarie.

Quando inserisci, aggiorni o cancelli i dati in un database devi sempre tenere conto dei vincoli.



CREARE UNA VISTA

Basandoti sulla tabella che hai creato con l'istruzione CREATE TABLE, puoi anche creare una tabella virtuale che esiste soltanto quando viene visualizzata da un utente. Queste tabelle si chiamano *viste*, e la tabella da cui una vista deriva si chiama *tabella di base*.



Usa l'istruzione SQL qui sotto per creare una vista.

```
CREATE VIEW prodotti_costosi  
(codice_prodotto,nome_prodotto,prezzo_unitario)  
AS SELECT *  
FROM prodotti  
WHERE prezzo_unitario>=200;
```

Quest'istruzione crea una vista.

La Tabella Prodotti Costosi è una vista basata sulla Tabella Prodotti, che rappresenta la tabella di base. È stata creata estraendo dalla Tabella Prodotti i record in cui il prezzo unitario è uguale o superiore a 200Au.

TABELLA PRODOTTI COSTOSI

Codice prodotto	Nome prodotto	Prezzo unit.
101	Melone	800Au
104	Limone	200Au
202	Caco	200Au

Una volta creata la vista dei Prodotti Costosi puoi ricercare al suo interno i dati con le stesse modalità con cui accederesti a una tabella base.

```
SELECT *  
FROM prodotti_costosi  
WHERE prezzo_unitario>=500;
```

Puoi usare una vista con le stesse modalità di una tabella base.

Può essere comodo creare una vista quando vuoi rendere pubblica solo una parte dei dati contenuti in una tabella.

Esistono anche istruzioni specifiche per cancellare una tabella base o una vista. Le mostriamo in due esempi.

```
DROP VIEW prodotti_costosi;
```

```
DROP TABLE prodotti;
```



DOMANDE

Crea le istruzioni SQL relative alle seguenti domande (l'unità di misura per la popolazione è 10.000 abitanti). Troverai le risposte a pagina 123.

D19

La Tabella Destinazioni qui sotto è stata creata usando l'istruzione CREATE TABLE. Aggiungi i dati qui sotto.

TABELLA DESTINAZIONI

Codice destinatario	Nome destinatario	Popolazione	Distretto
12	Regno di Minanmi	100	Mare del Sud
15	Regno di Paronu	200	Centrale
22	Regno di Tokanta	160	Mare del Nord
23	Impero Alfa	120	Mare del Nord

D20

Partendo dalla Tabella Destinazioni della Domanda 19, crea una vista chiamata *Nazioni del Mare del Nord* che contiene le nazioni appartenenti al distretto del Mare del Nord.

TABELLA DESTINAZIONI

Codice destinatario	Nome destinatario	Popolazione
22	Regno di Tokanta	160
23	Impero Alfa	120

D21

Modifica la popolazione del Regno di Tokanta nella Tabella Destinazioni portandola a 1,5 milioni.

D22

Cancella nella Tabella Destinazioni tutti i dati relativi al Regno di Paronu.

Riassumendo



- Puoi usare le funzioni di SQL per definire, gestire e controllare i dati.
- Per eseguire una ricerca utilizza l'istruzione SELECT.
- Per specificare una condizione utilizza la frase WHERE.
- Per inserire, aggiornare e cancellare i dati utilizza le istruzioni INSERT, UPDATE e DELETE.
- Per creare una tabella utilizza l'istruzione CREATE TABLE.

RISPOSTE

D1

```
SELECT *
FROM destinazioni
WHERE popolazione>=100;
```

D2

```
SELECT *
FROM destinazioni
WHERE popolazione<100;
```

D3

```
SELECT *
FROM destinazioni
WHERE codice_destinatario<20
AND popolazione>=100;
```

Codice destinatario	Nome destinatario	Popolazione
12	Regno di Minanmi	100

D4

```
SELECT *
FROM destinazioni
WHERE codice_destinatario>=30
AND popolazione>100;
```

Nessuno dei paesi soddisfa questi criteri e la query restituisce l'insieme vuoto.

Codice destinatario	Nome destinatario	Popolazione

D5

```
SELECT popolazione  
FROM destinazioni  
WHERE nome_destinatario='Regno di Ritol';
```

Popolazione
150

D6

```
SELECT *  
FROM destinazioni  
WHERE nome_destinatario LIKE '%n%';
```

Codice destinatario	Nome destinatario	Popolazione
12	Regno di Minanmi	100
30	Regno di Sazanna	80

D7

```
SELECT MIN(popolazione)  
FROM destinazioni;
```

MIN(popolazione)
80

D8

```
SELECT MAX(popolazione)  
FROM destinazioni;
```

MAX(popolazione)
300

D9

```
SELECT SUM(popolazione)  
FROM destinazioni;
```

SUM(popolazione)
1.350

D10

```
SELECT SUM(popolazione)
FROM destinazioni
WHERE codice_destinatario>20;
```

SUM(popolazione)
1.050

D11

```
SELECT COUNT(*)
FROM destinazioni
WHERE popolazione>=100;
```

COUNT(*)
7

D12

```
SELECT COUNT(*)
FROM destinazioni
WHERE distretto='Mare del Nord';
```

COUNT(*)
3

D13

```
SELECT MAX(popolazione)
FROM destinazioni
WHERE distretto='Mare del Nord';
```

MAX(popolazione)
240

D14

```
SELECT SUM(popolazione)
FROM destinazioni
WHERE NOT(nome_destinatario='Regno di Ritoli');
```

SUM(popolazione)
1.200

D15

```
SELECT distretto, AVG(popolazione)
FROM destinazioni
GROUP BY distretto
HAVING AVG(popolazione)>=200;
```

Distretto	AVG(popolazione)
Centrale	250

D16

```
SELECT distretto, COUNT(*)
FROM destinazioni
GROUP BY distretto
HAVING COUNT(*)>=3;
```

Distretto	COUNT(*)
Mare del Nord	3
Mare del Sud	3

D17

```
SELECT *
FROM risultati_vendita
WHERE codice_prodotto IN
(SELECT codice_prodotto
FROM prodotti
WHERE prezzo_unitario>=300);
```

D18

```
SELECT *
FROM risultati_vendita U
WHERE quantita<
(SELECT AVG(quantita)
FROM risultati_vendita
WHERE codice_prodotto=U.codice_prodotto);
```

N. registrazione	Codice prodotto	Quantità
1101	101	1.100
1102	103	1.700
1103	104	500

D19

```
INSERT INTO destinazioni(codice_destinatario,
nome_destinatario,popolazione,distretto)
VALUES(12,'Regno di Minanmi',100,'mare del sud');
INSERT INTO destinazioni(codice_destinatario,
nome_destinatario,popolazione,distretto)
VALUES(15,'Regno di Paronu',200,'centrale');
INSERT INTO destinazioni(codice_destinatario,
nome_destinatario,popolazione,distretto)
VALUES(22,'Regno di Tokanta',160,'mare del nord');
INSERT INTO destinazioni(codice_destinatario,
nome_destinatario,popolazione,distretto)
VALUES(23,'Impero Alfa',120,'mare del nord');
```

D20

```
CREATE VIEW nazioni_mare_nord(codice_destinatario,
nome_destinatario_popolazione)
AS SELECT codice_destinatario,nome_destinatario,popolazione
FROM nome_destinatario
WHERE distretto='mare del nord';
```

D21

```
UPDATE destinazioni
SET popolazione=150
WHERE nome_destinazione='Regno di Tokanta';
```

D22

```
DELETE FROM destinazioni
WHERE nome_destinatario='Regno di Paronu';
```

STANDARDIZZAZIONE DI SQL

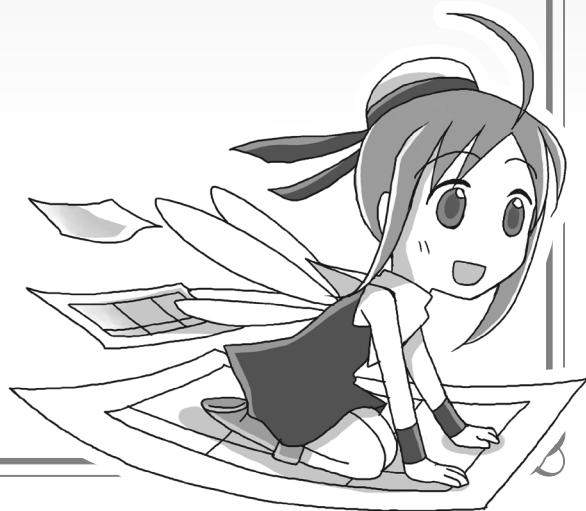
SQL risponde a uno standard dell'Organizzazione Internazionale per la Standardizzazione (ISO). In Giappone SQL è standardizzato da JIS (Japanese Industrial Standards).

Altri standard di SQL sono SQL92, approvato nel 1992, e SQL99, approvato nel 1999. I database relazionali sono progettati in modo che le query siano sempre conformi a questi standard.

Alcuni database relazionali, al contempo, presentano specifiche proprietarie. Fate riferimento al manuale operativo del vostro database per maggiori dettagli.

5

FACCIAMO FUNZIONARE UN DATABASE!



CHE COS'È UNA TRANSAZIONE?



PERÒ DEVO RINGRAZIARTI...

...ANCHE SE DOBBIAMO IMPARARE ANCORA MOLTO.

PER ESEMPIO, MI CHIEDO COME FACCIA A FUNZIONARE UN DATABASE QUANDO CI SONO COSÌ TANTI UTENTI COLLEGATI CONTEMPORANEAMENTE.

E IN EFFETTI, ANCHE LA QUESTIONE DELLA SICUREZZA MI PREOCCUPA.

SI DIREBBE CHE SIATE PREOCCUPATI PER IL DATABASE.

PIÙ O MENO.

BE', PER COMPRENDERE MEGLIO IL PROBLEMA...

...HO FATTO QUALCHE RICERCA.

OH, DAVVERO?

IL TITOLO DELLA MIA PRESENTAZIONE È:

COME FA UN DATABASE A CONSENTIRE L'ACCESSO DI TANTI UTENTI CONTEMPORANEAMENTE?

FLASH!

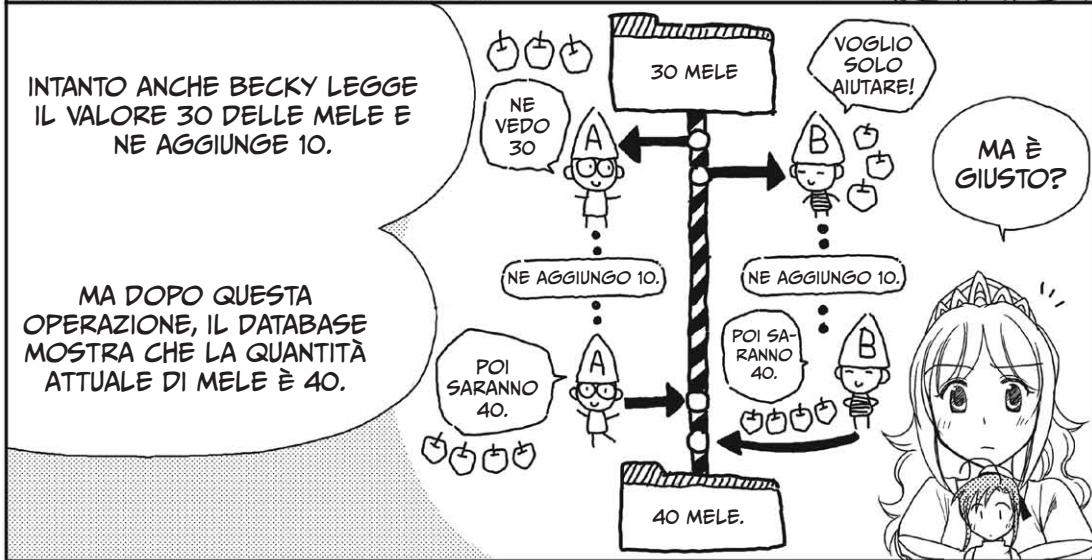
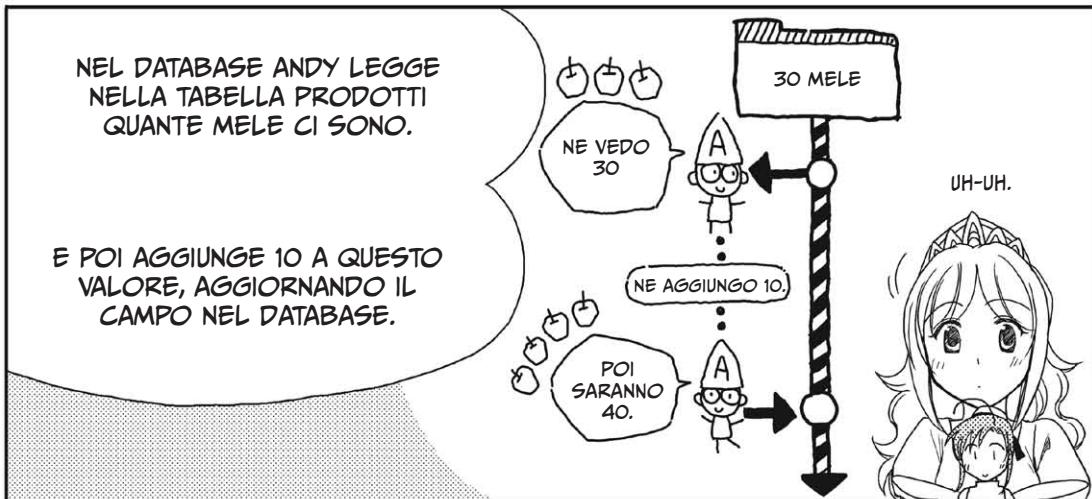
HO PREPARATO ANCHE DELLE ILLUSTRAZIONI PER AIUTARVI MEGLIO A CAPIRE!

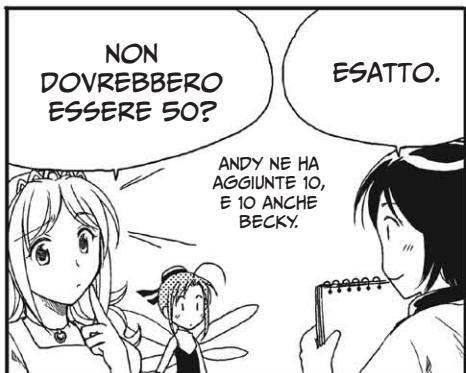
OH, MA È SPLENDIDO!

IL TEATRO DEL DATABASE

MI PIACCIONO GLI SPETTACOLI!

OH!





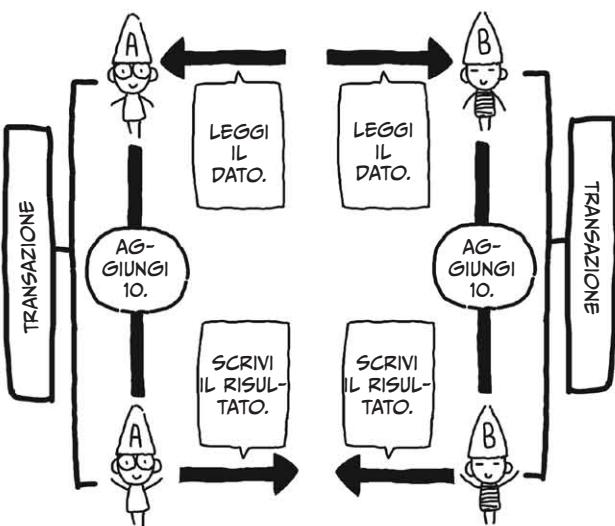
INNANZITUTTO,
UN DATABASE È
PROGETTATO
PER ELABORARE
LE OPERAZIONI
RAGGRUPPANDOLE.

GIÀGGIA.

UNA SEQUENZA INDIVISIBILE
DI OPERAZIONI SUI DATI VIENE
CHIAMATA **TRANSAZIONE**.

TRANSAZIONE?

NEL
NOSTRO ESEMPIO,
ABBiamo VISTO
UNA TRANSAZIONE
CHE COMPRENDE
UN'OPERAZIONE DI
LETTURA, DI ADDIZIONE
E DI SCRITTURA.



QUINDI LE
OPERAZIONI DI
ANDY FORMANO UNA
TRANSAZIONE,

MENTRE LE OPERAZIONI
DI BECKY NE FORMANO
UN'ALTRA.

CHE COS'È UN BLOCCO?

IN UN DATABASE
LE OPERAZIONI
DEI DIVERSI
UTENTI SONO
CONTROLLATE IN
MODO DA EVITARE
PROBLEMI...

...QUANDO GLI
UTENTI ACCEDONO
SIMULTANEALEMENTE.

CAPISCO.

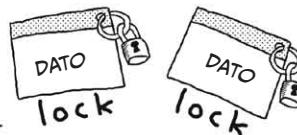
A QUESTO SCOPO
ESISTE UN METODO
DETTO **LOCK**, CHE
SIGNIFICA BLOCCO.



"LOCK" COME IL
"LUCCHETTO" DELLA
CHIAVE?

ESATTAMENTE.

PUOI BLOCCARE I
DATI PER EVITARE
CHE VENGANO
ELABORATI
ERRONEAMENTE.

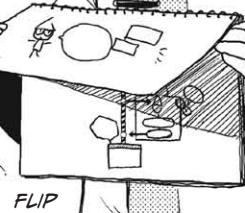


NON LO
SAPEVO.

PROVO A
SPIEGARMI
USANDO L'ESEMPIO
PRECEDENTE.

HAI
DISEGNATO
UN SACCO
DI SCHEMI!

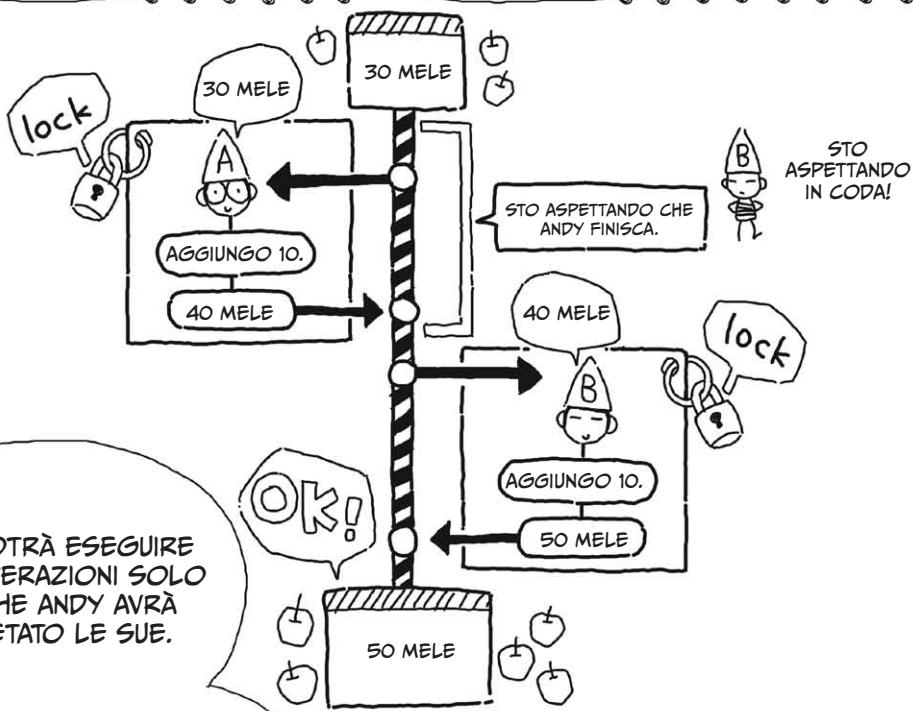
FLIP



FLIP

PRIMA DI ESEGUIRE UNA SERIE DI OPERAZIONI, ANDY BLOCCA I DATI.

QUANDO BECKY PROVA A ESEGUIRE LE SUE, DEVE PRIMA ATTENDERE CHE ANDY ABbia FINITO.

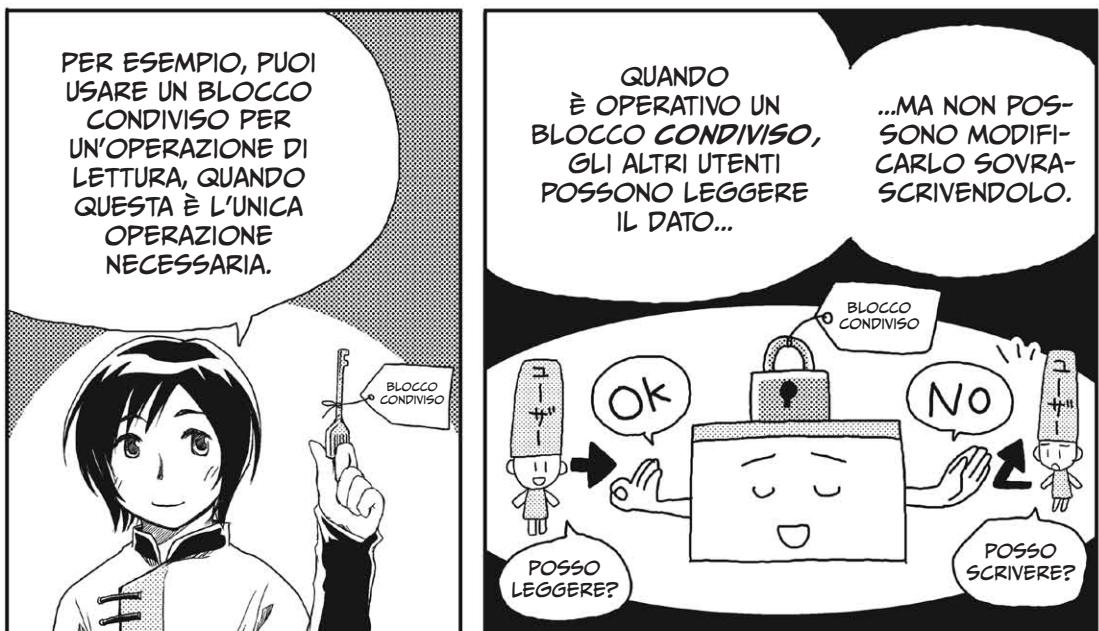
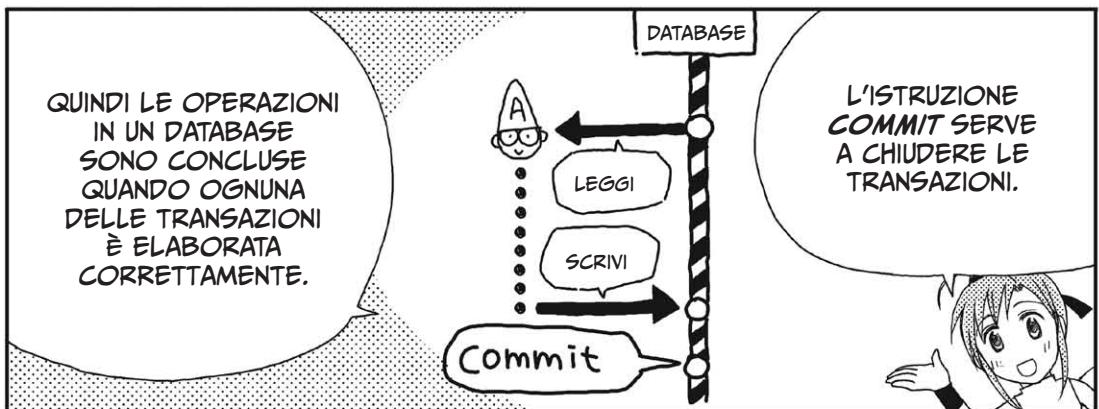


BECKY POTRÀ ESEGUIRE LE SUE OPERAZIONI SOLO DOPO CHE ANDY AVRÀ COMPLETATO LE SUE.

ALLA FINE NEL DATABASE TROVEREMO IL RISULTATO CORRETTO, CIOÈ 50 MELE.

RAGAZZI! CAIN, SONO COLPITA.

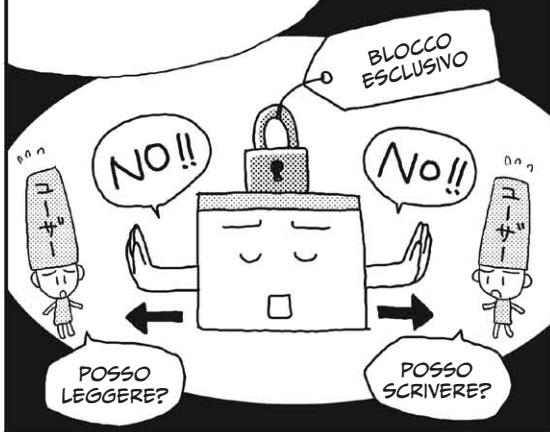




QUANDO UN UTENTE DEVE ESEGUIRE UN'OPERAZIONE DI SCRITTURA, ALLORA APPLICA UN BLOCCO ESCLUSIVO.



QUANDO È OPERATIVO UN BLOCCO ESCLUSIVO, GLI ALTRI UTENTI NON POSSONO NÉ LEGGERE NÉ SCRIVERE IL DATO.



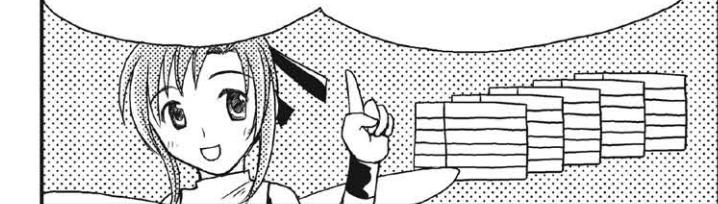
AH, CAPISCO, CI SONO TIPI DIVERSI DI BLOCCHI.

IN EFFETTO HA SENSO.

QUANDO UN BLOCCO VIENE USATO PER CONTROLLARE DUE O PIÙ TRANSAZIONI, SI PARLA DI CONTROLLO DI CONCORRENZA.

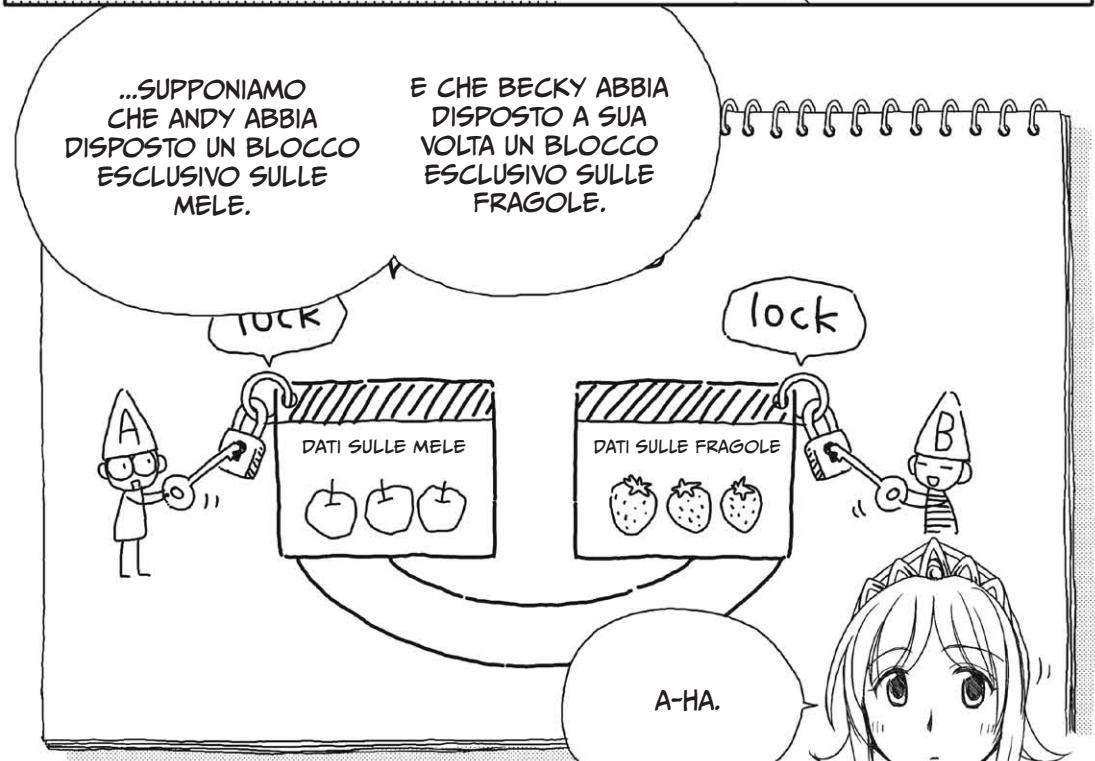
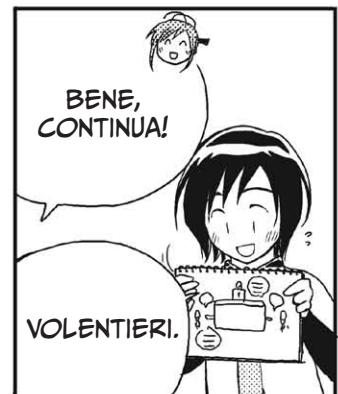
IN UN DATABASE, IL CONTROLLO DI CONCORRENZA PERMETTE...

...AL MAGGIOR NUMERO POSSIBILE DI UTENTI DI USARE IL DATABASE NELLO STESSO ISTANTE, EVITANDO CHE SI CREINO CONFLITTI.



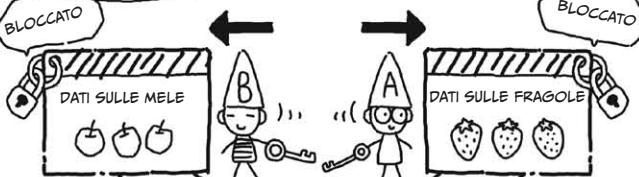
TRA PARENTESI...
CAIN!

SÌ, TICO?



POI ANDY PROVA
A DISPORRE UN
BLOCCO ESCLUSIVO
SULLE FRAGOLE...

...MENTRE BECKY
PROVA A DISPORRE
UN BLOCCO
ESCLUSIVO SULLE
MELE.



COSA
SUCCEDERÀ A
QUESTO PUNTO?

BE'...

VISTO CHE DEVONO
ENTRAMBI ASPETTARE
CHE IL BLOCCO
DELL'ALTRO SIA
RILASCIATO...

...NESSUNO
DEI DUE PUÒ
ESEGUIRE ALCUNA
OPERAZIONE. HO
DETTO BENE?

FAMMI
PENSARE...



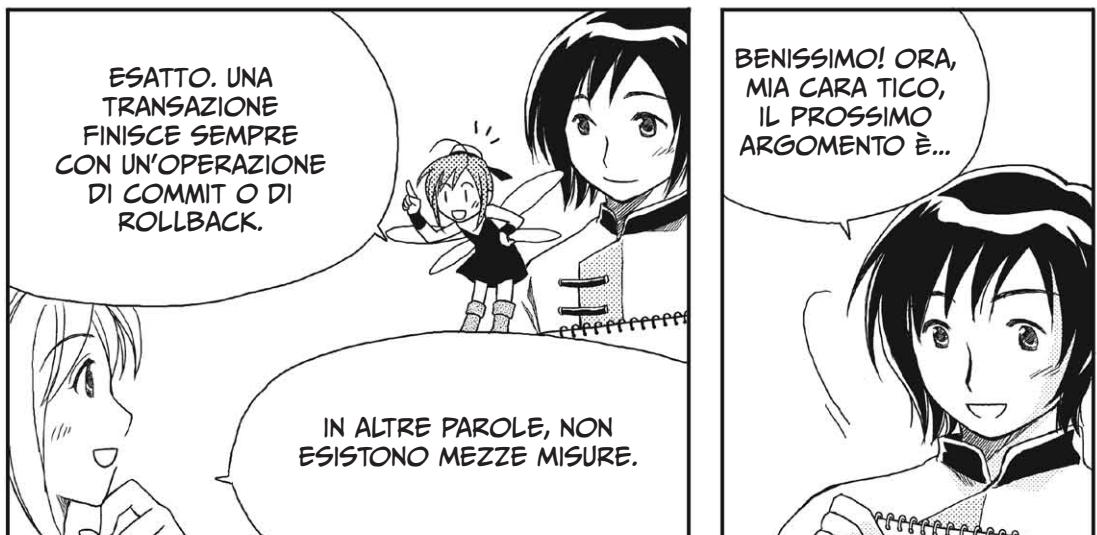
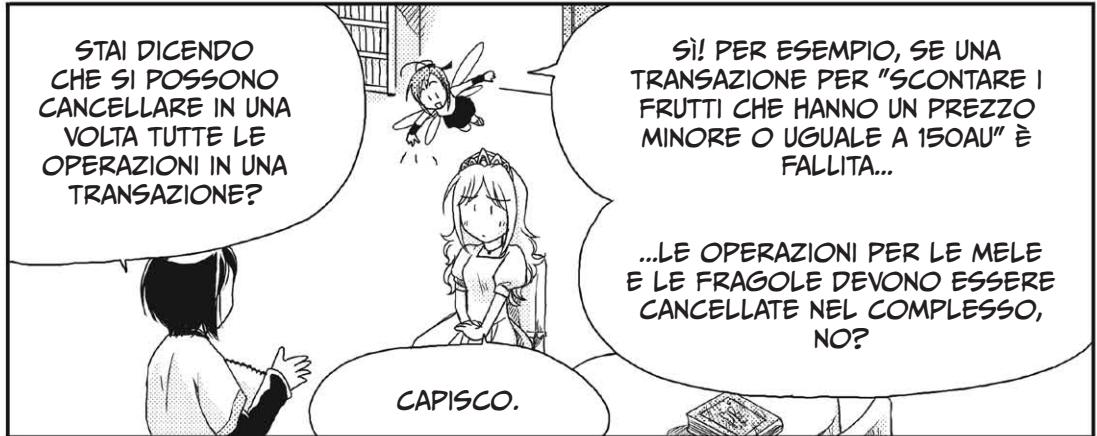
QUESTA
SITUAZIONE
PRENDE IL NOME
DI DEADLOCK, E
NON PUÒ ESSERE
RISOLTA FINCHÉ
UNO DEI BLOCCI
NON VIENE
RILASCIATO.

PER ESEMPIO, PUOI
CERCARE DELLE
TRANSAZIONI CHE
SONO RIMASTE
IN CODA PER UN
DETERMINATO
PERIODO...

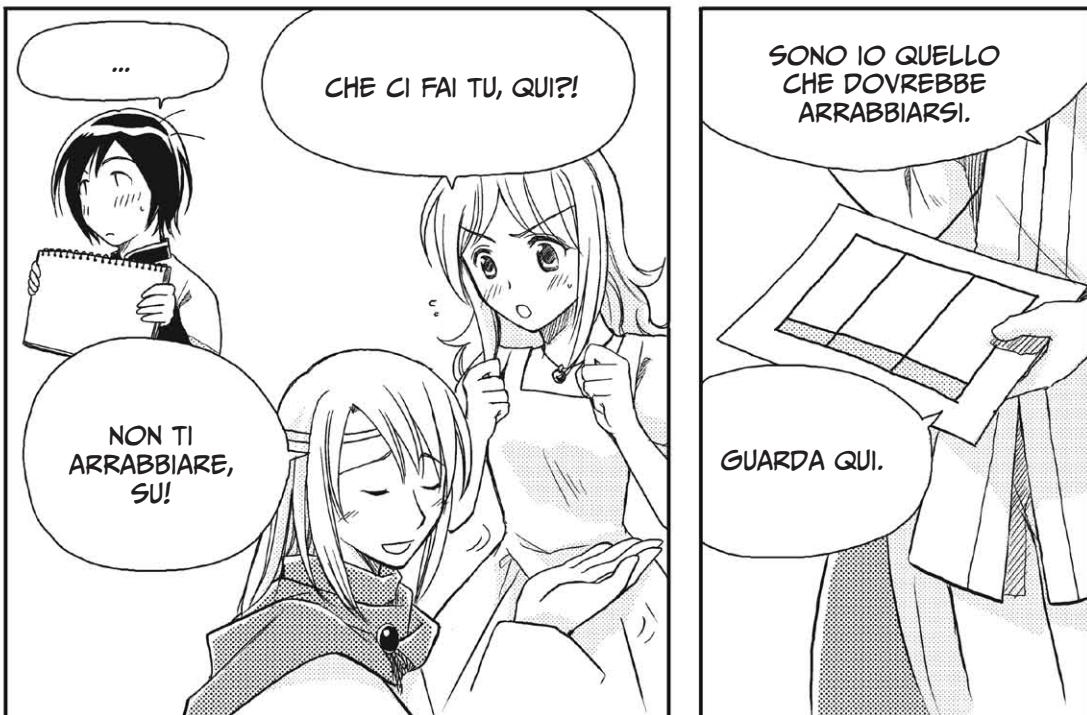
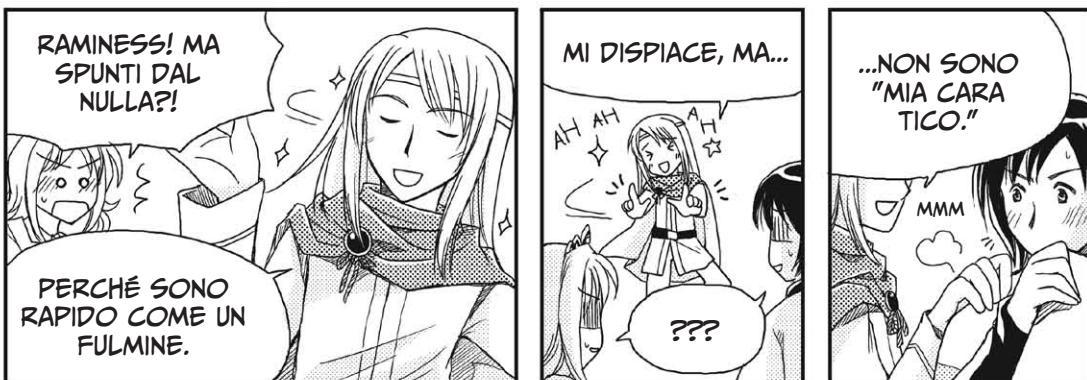
...E CANCELLARLE!

QUANDO CANCELLI
UNA TRANSAZIONE,
HAI ESEGUITO UN
ROLLBACK.

ROLLBACK?



SICUREZZA E DATABASE





CODICE PRODOTTO	NOME PRODOTTO	PREZZO UNITARIO
101	MELONE	10.000AU
102	FRAGOLA	12.500AU
103	MELA	8.000AU
104	LIMONE	6.000AU
201	CASTAGNA	9.000AU
202	CACO	12.400AU
301	PESCA	5.000AU
302	KIWI	6.000AU





PROMETTIAMO DI
MIGLIORARE LA
SICUREZZA DEL
NOSTRO DATABASE,
PER IMPEDIRE CHE UNA
COSA DEL GENERE SI
RIPETA DI NUOVO.

LA PREGO DI
PERDONARCI!

DICI CHE
SISTEMERETE TUTTO,
MA... NON SONO COSÌ
CONVINTO... PUOI
ESSERE UN PO' PIÙ
PRECISO?

IL PROBLEMA È
GENERATO DAL
FATTO CHE TUTTI
NEL REGNO DI KOD
HANNO LIBERO
ACCESSO AL
NOSTRO DATABASE.

PER PRIMA COSA
DOVREMO INSTAURARE
DEI CONTROLLI
D'ACCESSO, PER
LIMITARE IL NUMERO
DI UTENTI.

IL CHE
TRADOTTO...?

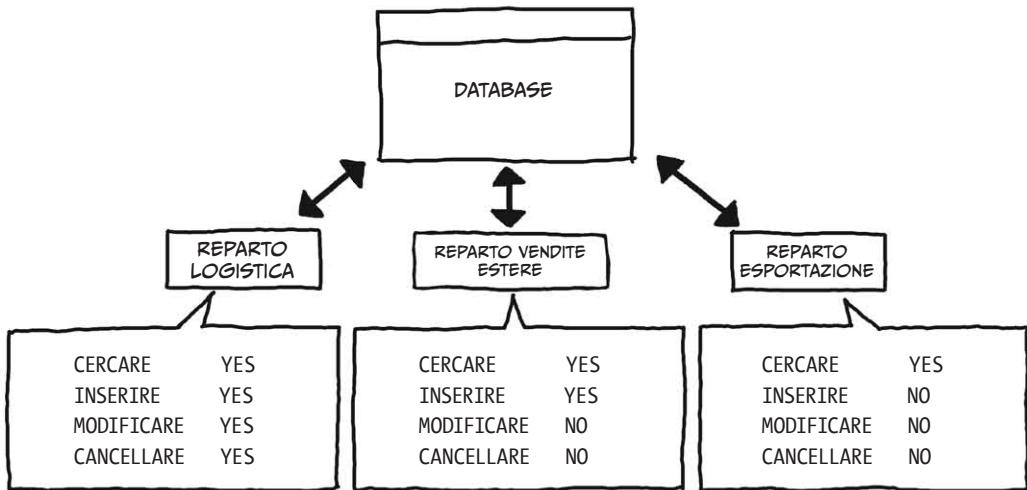
UNA BUONA SOLUZIONE
POTREBBE ESSERE
CONSENTIRE L'ACCESSO
SOLO TRAMITE NOME UTENTE
E PASSWORD, OGNI UTENTE
DOVRÀ DEMONSTRARE DI
ESSERE AFFIDABILE PER
POTER ENTRARE.

CHE IDEA
INTELLIGENTE!

POI CONFIGUREREMO
LE IMPOSTAZIONI
PER LIMITARE
CERTE OPERAZIONI
SOLO AGLI UTENTI
AUTORIZZATI A
FARLE.

- PERMESSO DI ESEGUIRE RICERCHE (SELECT),
INSERIRE, MODIFICARE E CANCELLARE I DATI
- PERMESSO DI ESEGUIRE RICERCHE, DI
INSERIRE I DATI MA NON DI MODIFICARLI O
CANCELLARLI
- PERMESSO DI ESEGUIRE RICERCHE...

- GLI OPERATORI DEL REPARTO LOGISTICA POSSONO ESEGUIRE RICERCHE, INSERIRE, AGGIORNARE E CANCELLARE DATI RELATIVI AI PRODOTTI.
- I VENDITORI ESTERI POSSONO ESEGUIRE RICERCHE E INSERIRE DATI RELATIVI AI PRODOTTI, MA NON POSSONO MODIFICARLI O CANCELLARLI.
- GLI ADDETTI ALL'ESPORTAZIONE POSSONO ESEGUIRE RICERCHE SUI PRODOTTI, MA NON INSERIRE, MODIFICARE O CANCELLARE I DATI.



はーん!!

NON LIMITEREMO
SOLTANTO IL NUMERO
DI UTENTI, STABILIREMO
ANCHE PER OGNI UTENTE
LE OPERAZIONI CHE È
AUTORIZZATO A
ESEGUIRE.

IN QUESTO MODO
EVITEREMO QUALSIASI
TIPO DI PROBLEMA, E
IL DATABASE RESTERÀ
CONDIVISO.

OH, SICURAMENTE...
MA PARLANDO
D'ALTRO, COME TI
STAVO DICENDO
PRIMA...

OH, MA
ASPETTA!

...MI SEMBRA
UN'OCCASIONE
IMPERDIBILE PER
ACCETTARE DI
SPOSARMI E...

SCUSS
←

L'INDICIZZAZIONE

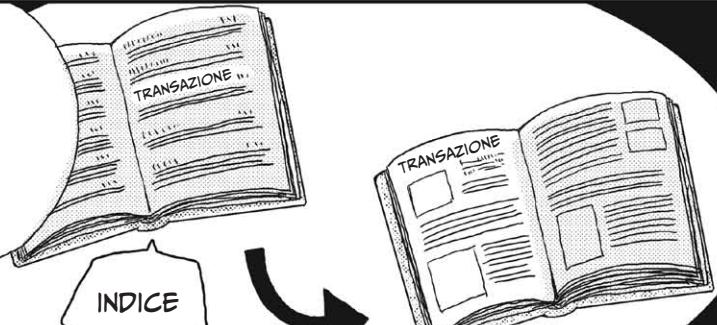


IMMAGINIAMO DI VOLER CERCARE IL SIGNIFICATO DELLA PAROLA **TRANSAZIONE** CONSULTANDO QUESTO LIBRO SUI DATABASE.

UNA RICERCA CASUALE SAREBBE DAVVERO COMPLICATA, QUINDI PER SEMPLIFICARCI LA VITA CONSULTIAMO L'INDICE.



ALLA VOCE **TRANSAZIONE** SONO RIPORTATE LE PAGINE IN CUI VIENE USATA QUELLA PAROLA.



USANDO UN INDICE SIAMO IN GRADO DI TROVARE AGEVOLMENTE LA PAGINA CHE CERCAVAMO!

PROPRIO COSÌ.

E L'INDICE IN UN DATABASE FUNZIONA ALLO STESSO MODO. PER ESEMPIO...



SE CREI INDICI PER I CODICI PRODOTTO...

...POTRAI SAPERE ISTANTANEAMENTE DOVE SI TROVANO I DATI RELATIVI A UN PRODOTTO A CUI È STATO ASSEGNATO IL CODICE PRODOTTO 101.



TI DICE DOVE SI TROVANO QUEI DATI NEL DISCO.

L'INDICIZZAZIONE AIUTA A VELOCIZZARE LE RICERCHE.

BE', FACCIO UN PO'
FATICA A SEGUIRTI,
MA...

UH-HUH.

DOVER TROVARE UN
DATO SCANDAGLIANDO
TUTTE LE RIGHE È
UNA VERA PERDITA DI
TEMPO.

VUOLE CHE
LE RIPETA DA
CAPO TUTTA LA
SPEGNAZIONE?

NO, NO...

Zio
du

USANDO GLI INDICI
POSSIAMO RIDURRE
GLI ACCESSI ALLA
MEMORIA.

E RIDUCENDO GLI
ACCESSI ALLA
MEMORIA, LE
RICERCHE SARANNO
PIÙ VELOCI!

CHE
SUCCIDE?

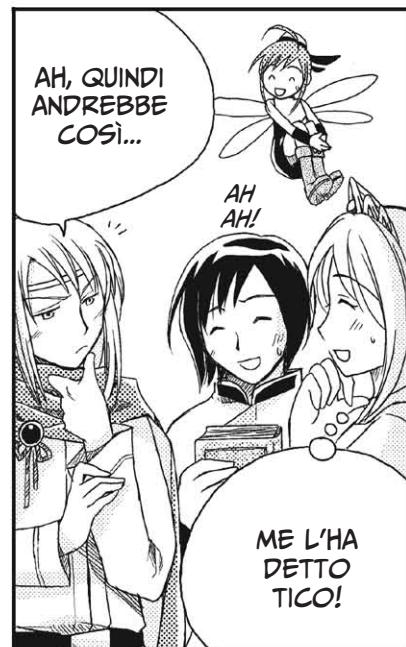


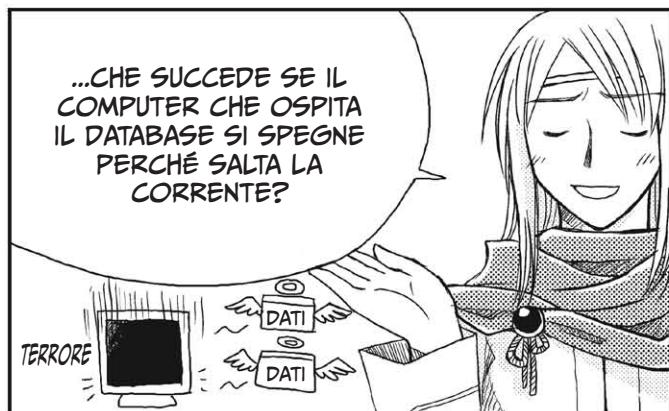
UH-OH.

EHI, CON
CHI STAI
PARLANDO?

C'È QUALCUNO
QUI?

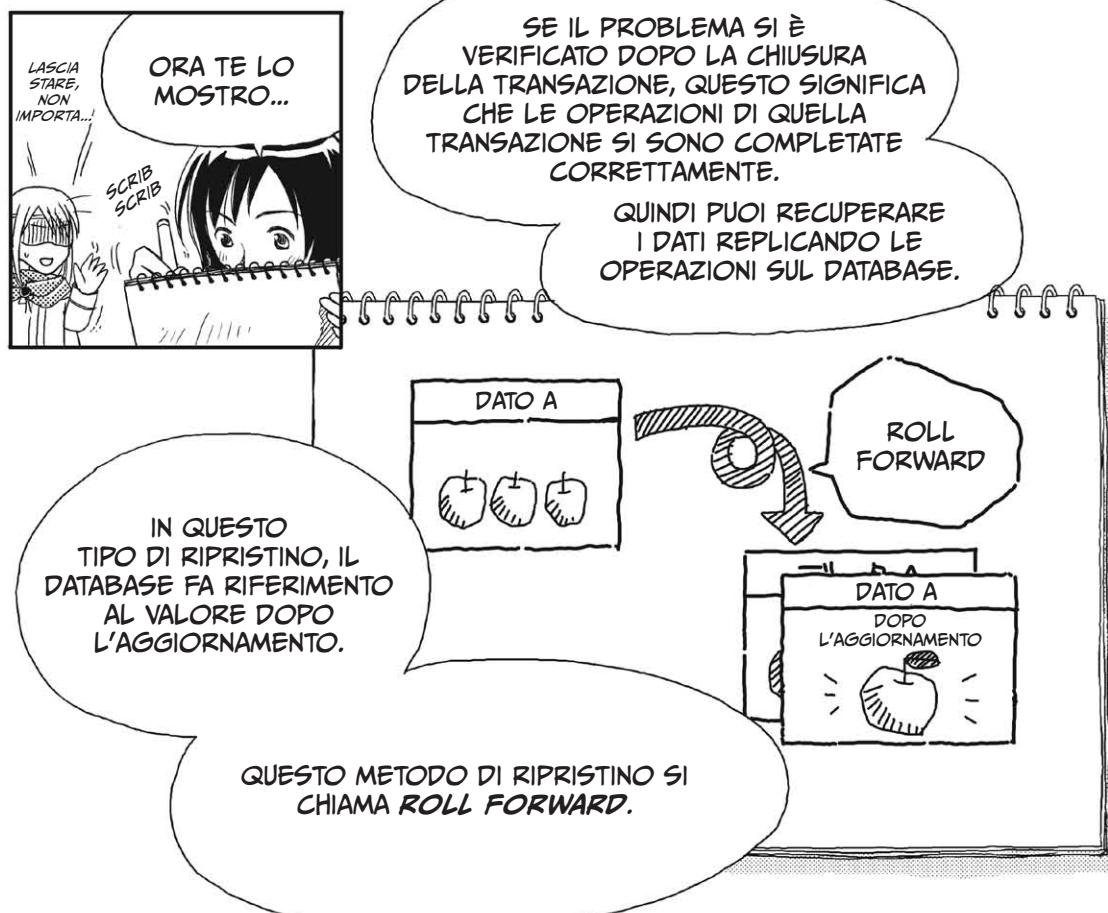
ATTENZIONE,
PRINCIPESSA!



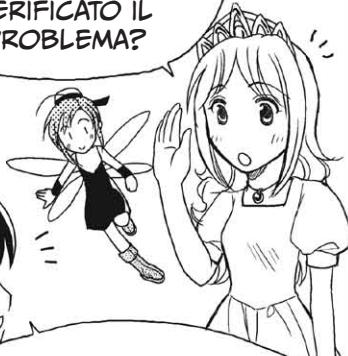


RIPRISTINO IN CASO DI MALFUNZIONAMENTO





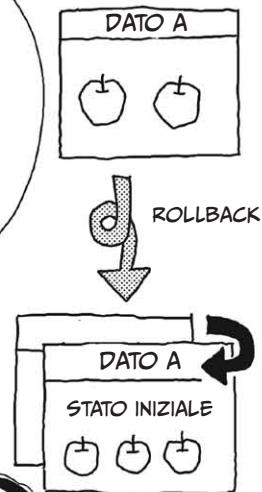
CHE SUCCIDE
INVECE SE LA
TRANSAZIONE NON
ERA CONCLUSA
QUANDO SI È
VERIFICATO IL
PROBLEMA?



NIENTE PAURA! IN QUEL
CASO ENTRA IN GIOCO IL
ROLLBACK.

QUANDO SI ESEGUE
UN ROLLBACK,
CI SI RIPORTA AI
VALORI PRECEDENTI
ALL'AGGIORNAMENTO,
PER CANCELLARE LA
TRANSAZIONE.

IN ALTRE PAROLE,
SI RIPRISTINA LO
STATO PRECEDENTE
ALL'INIZIO DELLA
TRANSAZIONE.



IL SISTEMA RIPRISTINA I
DATI E VERIFICA CHE NON
CI SIANO INCONGRUENZE.

UH-HUH.

TERMINI COME
TRANSAZIONE O
RIPRISTINO NON
MI SONO MOLTO
FAMILIARI.

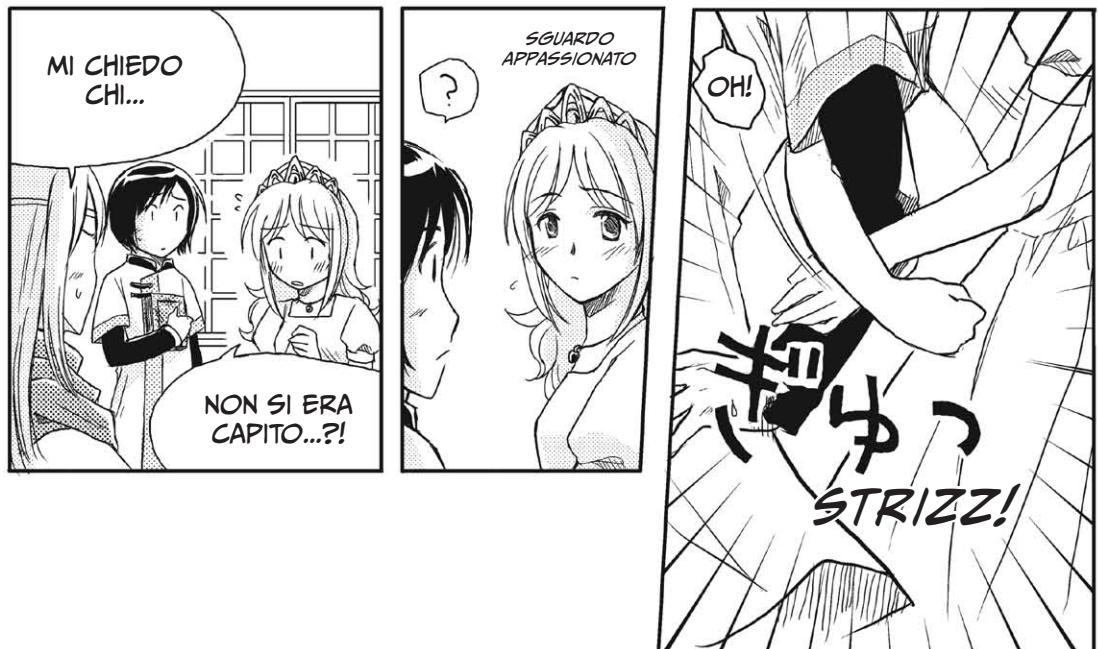
MM

MA DIREI CHE
LE MISURE DI
SICUREZZA DEL
VOSTRO DATABASE
SONO ALL'ALTEZZA.

SIGH

FINALMENTE
L'HA CAPITO!

VEDI, UN DATABASE
PUÒ E DEVE
RESISTERE ANCHE
IN PRESENZA DI
PROBLEMI!





RESTERÒ PER SEMPRE CON CAIN, E GRAZIE AL POTERE DEL NOSTRO DATABASE...



STAI DICENDO...



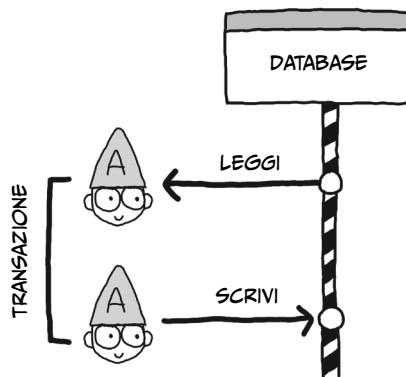
...FAREMOS FIORIRE E PROSPERARE IL REGNO DI KOD!



LE PROPRIETÀ DELLE TRANSAZIONI



La ricerca di Cain ha dimostrato che gli utenti di un database possono cercare, inserire, modificare e cancellare i dati. Un insieme di operazioni eseguite con successo da un singolo utente si chiama *transazione*.



Quando più utenti operano all'interno di un database è importante verificare che le transazioni concomitanti possano essere elaborate senza che si verifichino conflitti. È importante anche proteggere i dati da incongruenze, nel caso il sistema si blocchi nel corso di una transazione. A questo scopo, troverai nella tabella seguente una lista delle proprietà richieste in una transazione. Insieme formano l'acronimo *ACID*, semplice da memorizzare.

PROPRIETÀ RICHIESTE IN UNA TRANSAZIONE

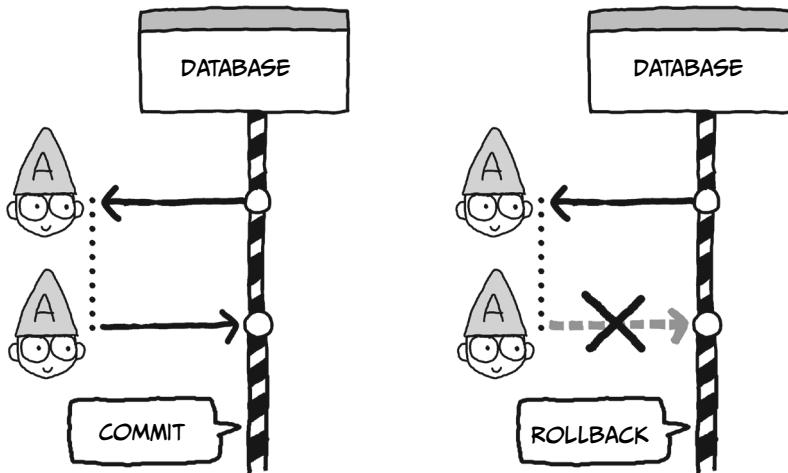
Proprietà	Iniziale di	Descrizione
A	Atomicità	Una transazione deve necessariamente chiudersi o essere annullata.
C	Coerenza	Una transazione non può mai causare una perdita di coerenza del database.
I	Isolamento	Anche quando più transazioni sono elaborate contemporaneamente, il loro risultato è identico a quello che avrebbero se elaborate in sequenza.
D	Durabilità	I valori risultanti da una transazione eseguita correttamente non possono essere persi per malfunzionamenti.

Esaminiamo in dettaglio ognuna di queste proprietà.

ATOMICITÀ

La prima proprietà necessaria in una transazione è l'*atomicità*, e implica che una transazione deve sempre terminare con un commit o con un rollback, per scongiurare la presenza di incongruenze nel database. Quindi, ognuna delle azioni di una transazione dev'essere completata correttamente, altrimenti tutte le azioni vengono cancellate.

L'istruzione *commit* segna la fine corretta di una transazione, mentre l'istruzione *rollback* cancella tutte le operazioni della transazione.



In certi casi il commit o il rollback vengono eseguiti automaticamente. Puoi persino specificare quale dei due dovrebbe essere eseguito. Per esempio, in presenza di errori puoi specificare l'esecuzione di un rollback.

Le istruzioni SQL COMMIT e ROLLBACK servono a eseguire queste operazioni.

<code>COMMIT;</code>	Usa quest'istruzione per confermare una transazione.
<code>ROLLBACK;</code>	Usa quest'istruzione per eseguire un ripristino.



DOMANDE

Rispondi a queste domande per verificare se hai compreso bene il concetto di atomicità. Troverai le risposte a pagina 167.

D1

Scrivi un'istruzione SQL che può essere utilizzata per chiudere una transazione.

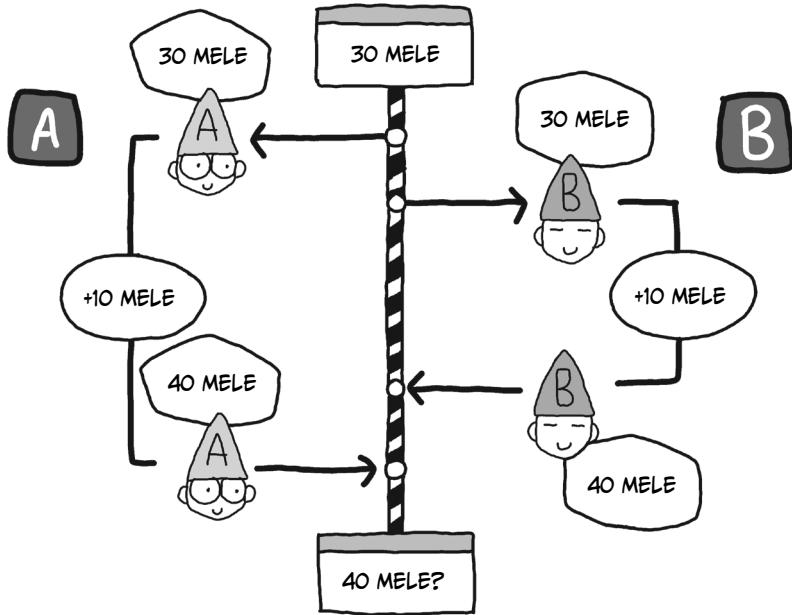
D2

Scrivi un'istruzione SQL che può essere utilizzata per cancellare una transazione.

COERENZA

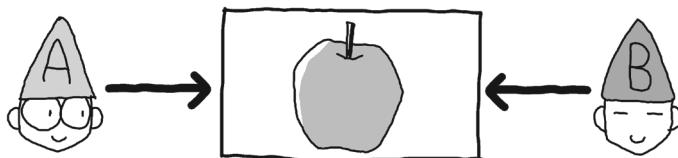
Una transazione non deve generare errori. Se il database era coerente prima dell'esecuzione di una transazione, allora dovrà esserlo anche una volta che questa è stata eseguita.

Cain ha fatto l'esempio di Andy e di Becky che cercano entrambi di aggiungere 10 mele a un totale di partenza di 30 mele. Il database non restituisce il risultato corretto (50 mele), ma ne mostra solo 40. Questo tipo di errori si chiama *perdita di aggiornamento*.



Quando due transazioni sono elaborate nello stesso momento può succedere che entrambe vogliano accedere allo stesso record della stessa tabella. Questo può causare conflitti nei dati.

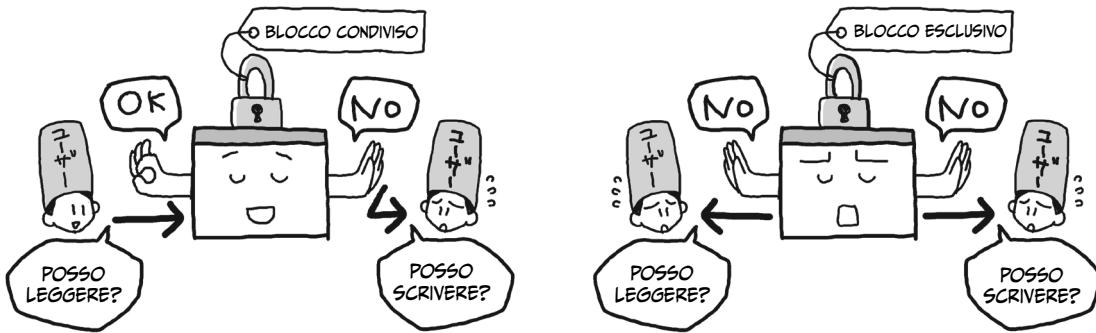
Le tabelle e le righe che durante una transazione sono soggette a operazioni sono chiamate *risorse*. In un database le transazioni devono essere in grado di accedere alla stessa risorsa contemporaneamente senza creare incoerenze.



ISOLAMENTO

Quando due o più transazioni vengono eseguite contestualmente e portano allo stesso risultato che avrebbero se eseguite singolarmente, si dice che la loro sequenza di operazioni è *serializzabile*. L'*isolamento* è una proprietà che richiede la serializzabilità, per proteggere il database dagli errori.

Per ottenere questa serializzabilità, devi poter controllare le transazioni che vengono effettuate nello stesso istante. Il metodo più comunemente usato a questo scopo è basato sui blocchi. Un *blocco condiviso* si usa quando i dati vengono solo letti, altrimenti si usa un *blocco esclusivo*.



Quando è in funzione un blocco condiviso, un altro utente può inserire blocchi condivisi ad altre transazioni, ma non blocchi esclusivi. Quando è in funzione un blocco esclusivo, nessun altro tipo di blocco può essere inserito. La tabella seguente riassume le relazioni possibili tra blocchi di tipo condiviso ed esclusivo.

RELAZIONI DI CO-ESISTENZA TRA BLOCCHI

	Blocco condiviso	Blocco esclusivo
Blocco condiviso	SI	NO
Blocco esclusivo	NO	NO



DOMANDE

Hai capito come funzionano i blocchi? Rispondi a queste domande, troverai le risposte a pagina 167.

D3

Se Andy ha applicato un blocco condiviso, Becky può applicare un blocco condiviso?

D4

Se Andy ha applicato un blocco esclusivo, Becky può applicare un blocco condiviso?

D5

Se Andy ha applicato un blocco condiviso, Becky può applicare un blocco esclusivo?

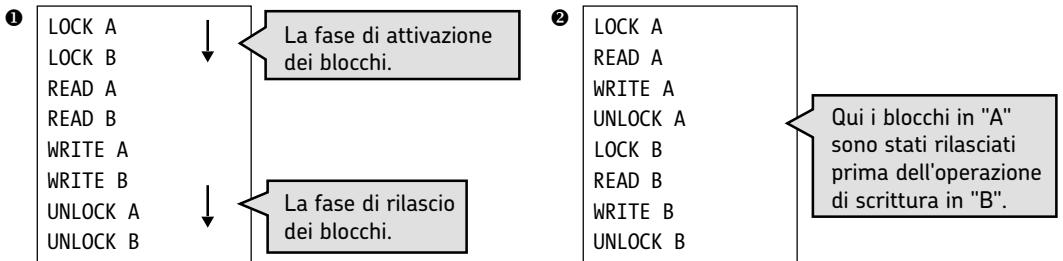
D6

Se Andy ha applicato un blocco esclusivo, Becky può applicare un blocco esclusivo?

IL BLOCCO A DUE FASI

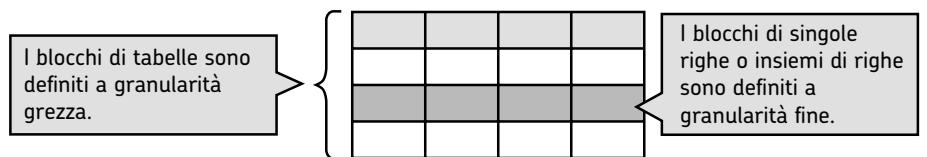
Per poter assicurare la serializzabilità di una sequenza di comandi, dobbiamo seguire delle regole ferree quando impostiamo o rilasciamo dei blocchi. Una di queste regole si chiama *blocco a due fasi*: per ogni transazione devono essere considerate due fasi: una in cui si applicano i blocchi e una in cui questi blocchi vengono rilasciati.

Per esempio, immaginiamo che due risorse A e B siano entrambe sottoposte a un blocco. La transazione ① osserva la regola del blocco a due fasi, a differenza della transazione ②. La serializzazione può essere ottenuta solamente quando tutte le transazioni seguono la regola del blocco a due fasi.



GRANULARITÀ DEI BLOCCHI

Esistono molti tipi di risorse che possono essere bloccate, per esempio puoi bloccare intere tabelle o intere righe di dati. L'estensione delle risorse bloccate viene anche chiamata *granularità*. Si definisce *grezza* quando implica il blocco di molte risorse allo stesso tempo, e *fine* quando invece comporta il blocco di poche risorse.



Quando la granularità è grezza (o alta) il numero di blocchi necessari per la transazione si riduce, rendendo più semplice la gestione della granularità. Al contempo si riducono anche il numero di operazioni della CPU su cui il database insiste. D'altro canto, visto che un numero maggiore di risorse viene bloccato, generalmente i tempi di attesa per rilasciare blocchi usati anche da altre transazioni è più lungo. Quindi, quando la granularità è alta, il numero di transazioni che si possono gestire contemporaneamente crolla.

Al contrario, quando la granularità è fine (o bassa) in una transazione viene eseguito un gran numero di blocchi, e questo richiede un maggior numero di operazioni per gestirli. Quindi le operazioni della CPU aumentano di conseguenza. Comunque, visto che i blocchi interessano meno risorse il rilascio dei blocchi richiederà in genere un tempo minore, rendendo possibile la gestione di un numero più alto di transazioni.

DOMANDE



Rispondi a queste domande, troverai le soluzioni a pagina 168.

D7

La risorsa destinataria di un blocco è stata modificata da una tabella a una riga. Cosa succederà al numero di transazioni che sarà possibile operare contemporaneamente?

D8

La risorsa destinataria di un blocco è stata modificata da una riga a una tabella. Cosa succederà al numero di transazioni che sarà possibile operare contemporaneamente?

ALTRI CONTROLLI DI CONCORRENZA

Puoi utilizzare i blocchi per eseguire con efficacia due o più transazioni allo stesso tempo. L'abuso dei blocchi, però, può portare a disagi nella gestione, visto il rischio di generare dei *deadlock*, o situazioni di stallo. In presenza di un numero limitato di transazioni o di un alto numero di operazioni di lettura ci sono sistemi più semplici di controllo della concorrenza. In questi casi specifici si possono usare i seguenti metodi:

Controllo sull'orario di accesso

Un'etichetta che contiene l'orario di accesso, chiamato in questo caso *timestamp*, viene assegnata ai dati coinvolti durante una transazione. Se un'altra transazione con un orario di accesso successivo ha già aggiornato i dati, l'operazione non verrà permessa. Quando un'operazione di lettura o di scrittura non viene permessa, la transazione si chiude con un rollback.

Controllo ottimistico

Questo metodo consente le operazioni di lettura. Quando viene tentata una scrittura, viene controllata la presenza di altre transazioni occorse sul dato. Se un'altra transazione ha già aggiornato il dato, la transazione si chiude con un rollback.

LIVELLI DI ISOLAMENTO

Nell'uso comune puoi impostare il livello di transazioni che possono essere elaborate nello stesso tempo. Questa impostazione si chiama *livello di isolamento*.

In SQL l'istruzione SET TRANSACTION può essere usata per specificare i livelli di isolamento delle transazioni seguenti:

- READ UNCOMMITTED
- READ COMMITTED
- REPEATABLE READ
- SERIALIZABLE

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
```

A seconda del livello di isolamento impostato si possono ottenere i seguenti livelli di protezione.

	Lettura sporca	Lettura non ripetibile	Lettura fantasma
READ UNCOMMITTED	Possibile	Possibile	Possibile
READ COMMITTED	Non può verificarsi	Possibile	Possibile
REPEATABLE READ	Non può verificarsi	Non può verificarsi	Possibile
SERIALIZABLE	Non può verificarsi	Non può verificarsi	Non può verificarsi

- Una *lettura sporca* si verifica quando la transazione 2 legge una riga prima che la transazione 1 sia chiusa con un commit.
- Una *lettura non ripetibile* si verifica quando una transazione legge lo stesso dato due volte ottenendo un valore diverso.
- Una *lettura fantasma* si verifica quando una transazione cerca le righe che corrispondono a una determinata condizione, ma trova le righe sbagliate per colpa delle modifiche apportate da un'altra transazione.

DURABILITÀ

Un database può gestire dati importanti, quindi è fondamentale poter garantire la sua sicurezza e la persistenza dei dati in caso di problemi. La sicurezza è importante anche quando si tratta di impedire a utenti non autorizzati di modificare i dati (generando problemi di coerenza).

In un database puoi stabilire i diritti di accesso di tutti gli utenti. Aumentandone il livello di sicurezza, Cain ha scongiurato possibili pericoli al database del Regno.

In un database relazionale, l'istruzione GRANT permette di assegnare i permessi di lettura e scrittura agli utenti. Puoi usare quest'istruzione per regolare l'accesso degli utenti alle tabelle che hai creato. Assegnare i permessi è un'operazione molto importante nella gestione di un database.

```
GRANT SELECT, UPDATE ON prodotti TO reparto_vendite_estere;
```

Quest'istruzione assegna il permesso di elaborare i dati.

Con SQL puoi assegnare i seguenti privilegi (permessi).

PRIVILEGI SU DATABASE

Istruzione	Risultato
SELECT	Permette agli utenti di cercare le righe in una tabella
INSERT	Permette agli utenti di aggiungere righe a una tabella
UPDATE	Permette agli utenti di aggiornare le righe di una tabella
DELETE	Permette agli utenti di cancellare righe in una tabella
ALL	Assegna tutti i privilegi

Se aggiungi l'opzione WITH GRANT OPTION, stai consentendo all'utente di assegnare privilegi ad altri utenti. Grazie all'istruzione seguente, l'Ufficio Vendite Estere può permettere ad altri utenti di eseguire ricerche e aggiornare il database.

```
GRANT SELECT, UPDATE ON prodotti TO reparto_vendite_estere  
WITH GRANT OPTION;
```

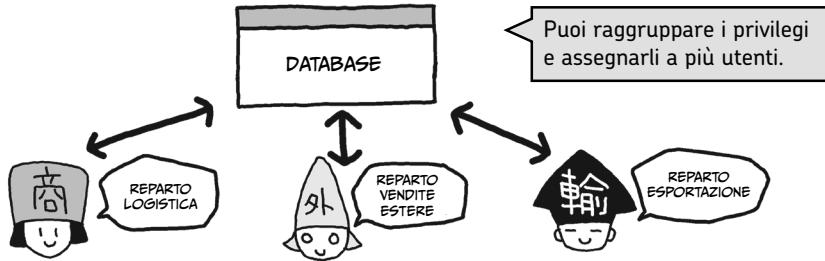
L'assegnatario può assegnare privilegi ad altri utenti.

Puoi anche revocare i privilegi di un utente, usando l'istruzione REVOKE.

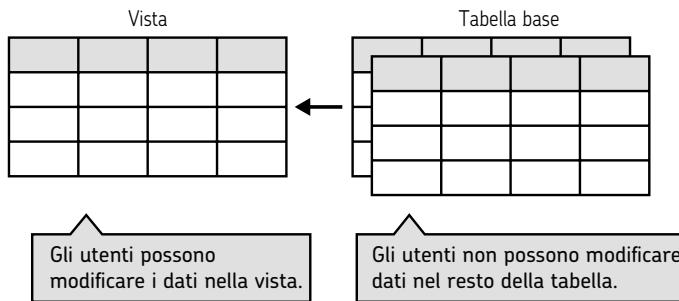
```
REVOKE SELECT, UPDATE ON prodotti FROM  
reparto_vendite_estere;
```

Quest'istruzione revoca i privilegi dati all'utente.

Alcuni applicativi permettono di raggruppare i privilegi e assegnarli a più utenti contemporaneamente. Questa possibilità rende più semplice gestire i privilegi.



Grazie alle viste, come descritte a pagina 117, hai accesso a una gestione ancora più controllata e sicura. Per prima cosa, estrai parte di una tabella base per generare una vista. Assegnare un privilegio per questa vista sottintende che il privilegio è assegnato anche nella porzione selezionata di dati che contiene.



DOMANDE

Prova a rispondere a queste domande sulla durabilità. Troverai le risposte a pagina 168.

D9

Scrivi un'istruzione SQL che consenta al Reparto Esportazione di cercare dati nella Tabella Prodotti.

D10

Crea un'istruzione SQL per revocare al Reparto Vendite Estere il privilegio di cancellare dati dalla Tabella Prodotti.

D11

L'amministratore della Tabella Prodotti ha assegnato i seguenti privilegi. Completa la tabella seguente con SÌ o NO per indicare l'assenza o la presenza di un privilegio in ognuno dei reparti indicati.

```
GRANT ALL prodotti TO reparto_vendite_estere;
GRANT INSERT, DELETE ON prodotti TO reparto_logistica;
GRANT UPDATE, DELETE ON prodotti TO reparto_esportazioni;
```

	Select	Insert	Update	Delete
Reparto Vendite Estere				
Reparto Logistica				
Reparto Esportazione				

IN CASO DI DISASTRO



Un database deve prevedere un meccanismo di protezione dei dati, nel caso si verifichi un problema (fallimento). Per assicurare la durabilità delle transazioni è necessario impedire che un fallimento possa generare dati non corretti o incompleti. Per proteggersi da queste eventualità il database ha a disposizione diverse operazioni, che includono la creazione di backup e di registri delle transazioni.

LE TIPOLOGIE DI FALLIMENTO

Un fallimento può avvenire in diverse circostanze, tra le più comuni ci sono le seguenti:

- Fallimento di transazione (Transaction failure)
- Fallimento di sistema (System failure)
- Fallimento fisico (Media failure)

Una *transaction failure* avviene quando una transazione non può essere completata per via di un errore nella transazione stessa. In questi casi viene eseguito un ripristino (rollback).

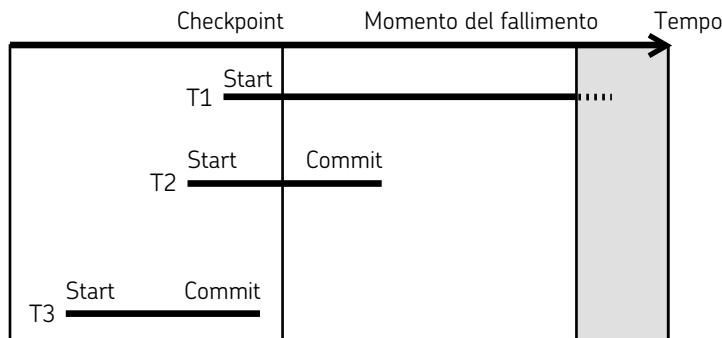
Una *system failure* avviene quando il sistema si spegne, per esempio a causa di un'interruzione dell'alimentazione. Quando succedono fallimenti di questo tipo, l'operazione di recupero prende il via subito dopo il reboot del sistema. In genere le transazioni che non erano state ancora chiuse al momento del fallimento vengono perdute (rollback), mentre quelle che erano già state completate al momento del fallimento vengono confermate (roll forward).

Una *media failure* avviene quando il disco che contiene i database è danneggiato. Nel caso di un tale fallimento, il ripristino è eseguito utilizzando i backup. Per le transazioni finalizzate dopo che i file di backup erano stati creati sarà eseguito un roll forward.

I CHECKPOINT

Al fine di migliorare l'efficienza di un'operazione di scrittura in un database, spesso si usa un *buffer* (un segmento della memoria predisposto a memorizzare dati temporanei). Nel momento in cui il contenuto del buffer è sincronizzato a quello del database viene fissato un *checkpoint*. Quando il database scrive un checkpoint non deve effettuare nessun ripristino per tutte le transazioni finalizzate prima del checkpoint stesso. Le transazioni che non si erano chiuse prima del checkpoint dovranno invece essere recuperate.

Ora, supponiamo che mentre sono in esecuzione le transazioni mostrate qui sotto, avvenga un fallimento di sistema. Quali transazioni dovrebbero essere gestite con un rollback? Quali con un roll forward?





DOMANDE

Queste domande fanno riferimento alla tabella nella pagina precedente. Troverai le risposte a pagina 168.

D12

Come dev'essere processata la transazione T1?

D13

Come dev'essere processata la transazione T2?

D14

Come dev'essere processata la transazione T3?

In caso di fallimento, i meccanismi di ripristino descritti in precedenza proteggeranno il database dal rischio di incongruenze nei dati. L'integrità di un database è un prerequisito fondamentale per gli utenti.

GLI INDICI

Un database gestisce grandi quantità di dati, quindi la ricerca di uno specifico dato può risultare molto laboriosa. Ma gli indici consentono di accelerare le ricerche!

Codice prodotto	Nome prodotto	Prezzo unit.	Distretto
101	Melone	800Au	Mare del Sud
102	Fragola	150Au	Centrale
103	Mela	120Au	Mare del Nord
104	Limone	200Au	Mare del Sud
201	Castagna	100Au	Mare del Nord
202	Caco	160Au	Centrale
301	Pesca	130Au	Mare del Sud
302	Kiwi	200Au	Mare del Sud

Ricercare un dato una riga alla volta richiede molto tempo!

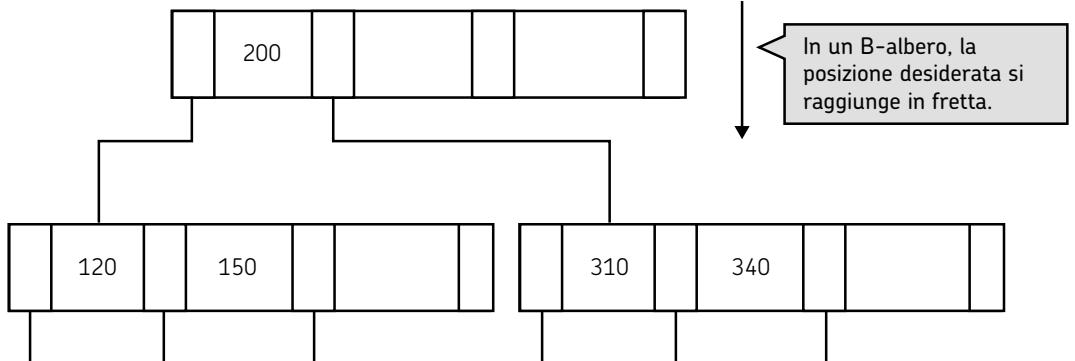
Un *indice* è uno strumento che consente di accedere velocemente alla posizione del dato ricercato. Quando si esegue una ricerca in database molto grandi, l'uso degli indici consente di trovare in fretta il dato desiderato.

Codice prodotto	Nome prodotto	Prezzo unit.	Distretto
101	Melone	800Au	Mare del Sud
102	Fragola	150Au	Centrale
103	Mela	120Au	Mare del Nord
104	Limone	200Au	Mare del Sud
201	Castagna	100Au	Mare del Nord
202	Caco	160Au	Centrale
301	Pesca	130Au	Mare del Sud
302	Kiwi	200Au	Mare del Sud

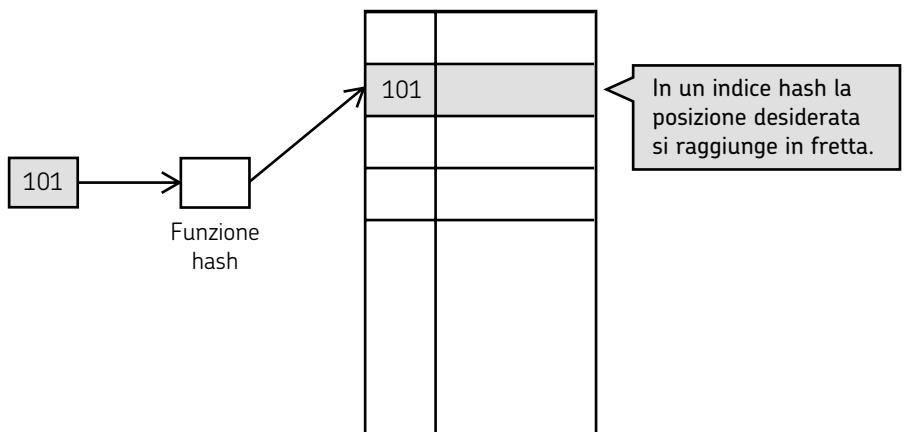
La posizione del dato ricercato è raggiungibile velocemente usando il suo indice.

Index

Esistono vari metodi di indicizzazione, tra cui i B-alberi e le tabelle hash. Un *B-albero* si compone di nodi padre e di nodi figlio, i quali possono a loro volta avere dei nodi figlio. I nodi sono disposti in ordine, ogni nodo padre contiene informazioni sul valore minimo e sul valore massimo contenuto da tutti i suoi figli. Questo permette al database di raggiungere in fretta la posizione desiderata, saltando intere sezioni dell'albero che sicuramente non contengono il valore desiderato.



L'indice *hash* invece consente di trovare la posizione del dato ricercato applicando una funzione hash al valore chiave del dato. L'hash è unico, come le impronte digitali. Questo metodo permette di eseguire ricerche molto specifiche, come per esempio la ricerca del *codice prodotto 101*. Non è però progettato per funzionare efficacemente nei casi in cui la ricerca usi condizioni comparative come *codice prodotto maggiore di 101* o condizioni poco definite come *prodotti che finiscono per 'e'*.



In certi casi usare un indice potrebbe non semplificare una ricerca: un indice non permette di guadagnare tempo, a meno che la ricerca non sia circoscritta a una piccola porzione dei dati. Esistono anche casi in cui gli indici devono essere ricreati ogni volta che viene modificato un dato, e questo provoca un rallentamento nel processare un aggiornamento.



DOMANDE

Verifica la tua comprensione degli indici. Troverai le risposte a pagina 168.

D15

Quale indice sarebbe più efficace se la tua condizione fosse 'uguale a...', un B-albero o un indice hash?

D16

Quale indice sarebbe più efficace se la tua condizione fosse 'diverso da...', un B-albero o un indice hash?

OTTIMIZZARE UNA QUERY

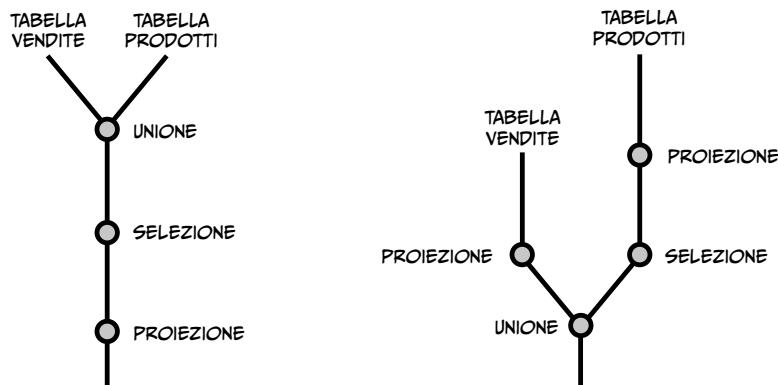
Quando esegui una query, il database analizza l'istruzione SQL e considera se è opportuno usare un indice per elaborarla più speditamente. Esaminiamo la procedura con cui viene elaborata una query.

Il database può decidere qual è l'ordine ottimale per elaborare una query. Molte query possono essere processate variando l'ordine delle istruzioni senza che questo influisca sul risultato, ma con benefici sulla velocità di ricerca. Per esempio, immaginiamo una query che serve a estrarre le date delle vendite e i nomi dei prodotti caratterizzati da un prezzo unitario superiore a 200Au. Questa query può essere riassunta nei passi seguenti:

```
SELECT data, nome_prodotto  
FROM prodotti, vendite  
WHERE prezzo_unitario>=200  
AND prodotti.codice_prodotto = vendite.codice_prodotto;
```

1. Unisci la Tabella Prodotti e la Tabella Vendite.
2. Seleziona i prodotti il cui prezzo unitario è maggiore o uguale a 200Au.
3. Estrai le colonne data e nome prodotto.

Lo schema qui sotto a sinistra mostra la query elaborata nell'ordine da 1 a 3. Lo schema a destra, invece, mostra la query elaborata nell'ordine da 3 a 1. Il risultato di entrambe le query è lo stesso.



A prescindere dal risultato, procedere da 1 a 3 richiede in generale un tempo di elaborazione più lungo, perché quando viene realizzata la prima unione si genera una tabella intermedia che contiene moltissime righe. Invece, procedere da 3 a 1 richiede meno tempo, perché la selezione e la proiezione provvedono a escludere dati non desiderati sin dal principio. Quindi una medesima query può richiedere tempi di elaborazione diversi, a seconda dell'ordine in cui l'unione, la selezione e la proiezione vengono eseguite.

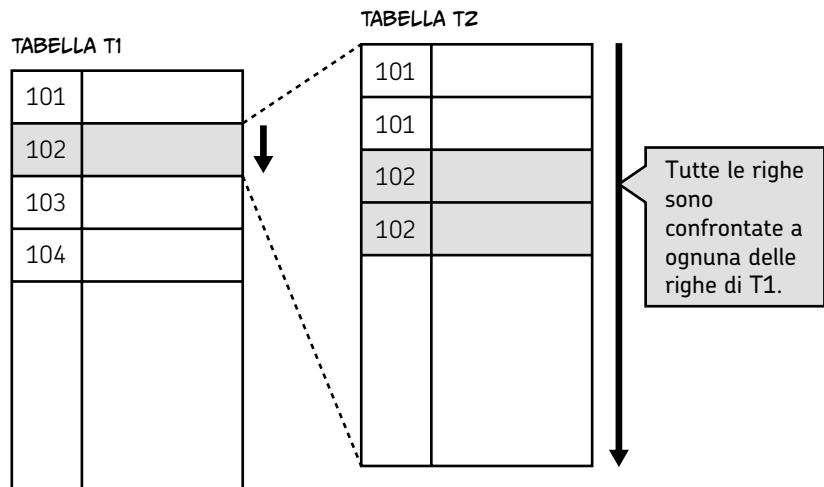
In genere, il database dovrebbe seguire queste regole per stabilire l'ordine di esecuzione ottimale:

- Eseguire prima la selezione, per ridurre il numero di righe.
- Eseguire prima la proiezione, per ridurre il numero di colonne non rilevanti al risultato.
- Posticipare l'unione.

Esistono diverse tecniche per eseguire proiezione, selezione e unione. Per la selezione si possono avere ricerche per concordanza totale (full-match) o ricerche basate su indici. Per l'unione sono disponibili i metodi seguenti.

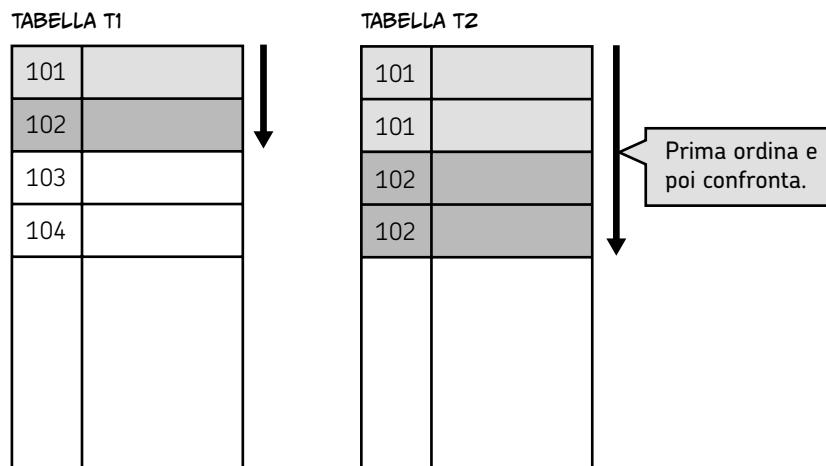
I CICLI ANNIDATI

Il metodo dei *cicli annidati* confronta una riga in una tabella a diverse righe in un'altra tabella (vedi la figura qui sotto). Per esempio, uno dei valori in una riga della Tabella T1 viene usato per trovare righe equivalenti nella Tabella T2. Se i valori sono identici, allora vengono unite le righe.



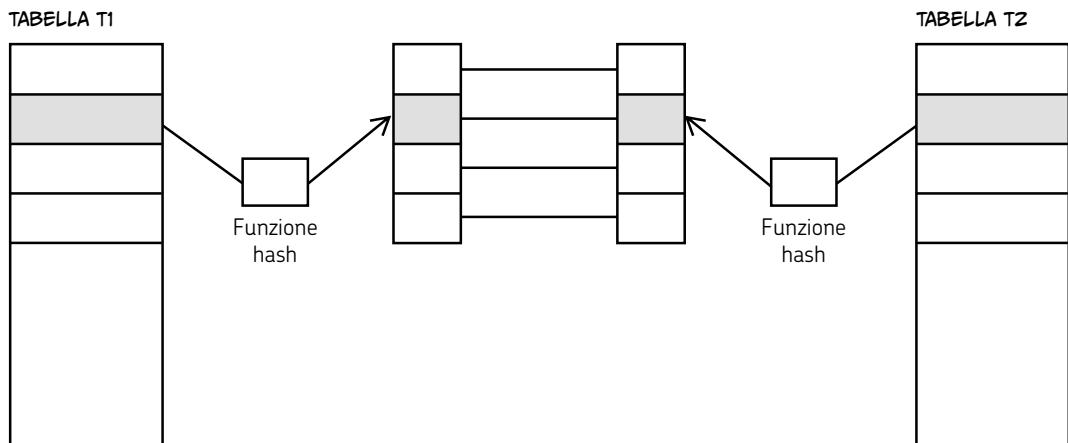
SORT MERGE (ORDINA E UNISCI)

Il metodo *sort merge* ordina e unisce righe in tabelle separate (vedi la figura sotto). Prima, una porzione o tutte le Tabelle T1 e T2 vengono ordinate. Poi sono confrontate a partire dalla prima riga, e tutte le volte che viene trovato un valore identico le righe vengono unite. Visto che le tabelle sono state pre-ordinate il meccanismo deve lavorare in una sola direzione, con evidente risparmio di tempo. Bisogna ovviamente considerare il tempo necessario all'inizio per ordinare le righe.



HASH

Un *hash* divide una delle tabelle utilizzando una funzione hash, e poi le collega con una riga in un'altra tabella con lo stesso valore di hash. Questo metodo seleziona efficacemente le righe che devono essere unite.



L'OTTIMIZZATORE

Quando viene eseguita una query, ci sono diverse tecniche per ottimizzare la sua performance. In un database, la funzione deputata all'ottimizzazione delle query si chiama *ottimizzatore*. Gli ottimizzatori sono di due tipi.

OTTIMIZZAZIONE BASATA SULLE REGOLE

Prima di eseguire le operazioni vengono stabilite alcune regole. Per esempio, alcune operazioni possono essere combinate o rimescolate, così come spesso in un'espressione algebrica gli operatori possono cambiare di posto senza modificare il risultato. L'ottimizzatore cerca il modo più efficiente di elaborare la query senza modificarne il risultato.

OTTIMIZZAZIONE BASATA SUI COSTI

Questo metodo cerca di stimare quanto costa in termini di tempo eseguire la query, sulla base di dati statistici raccolti dal database. La gestione *basata sui costi* può risultare più flessibile di quella basata sulle regole, ma richiede un aggiornamento periodico dei dati statistici da parte del database. La gestione e l'analisi di questi dati statistici richiede molto tempo.

RIASSUMENDO



- Puoi stabilire i privilegi per gli utenti del tuo database.
- I blocchi proteggono la coerenza dei dati quando il database ha molti utenti.
- L'indicizzazione permette di velocizzare le ricerche.
- Un database possiede meccanismi di ripristino in caso di fallimento.

RISPOSTE

D1

COMMIT;

D2

ROLLBACK;

D3 Sì.

D4 No.

D5 No.

D6 No.

D7 Aumenta.

D8 Diminuisce.

D9

```
GRANT SELECT ON prodotti TO reparto_esportazioni;
```

D10

```
REVOKE DELETE ON prodotti FROM reparto_vendite_estere;
```

D11

	Select	Insert	Update	Delete
Reparto Vendite Estere	Sì	Sì	Sì	Sì
Reparto Logistica	NO	Sì	NO	Sì
Reparto Esportazione	NO	NO	Sì	Sì

D12

Con un rollback, visto che la transazione non si era chiusa con un commit nel momento del fallimento.

D13

Con un roll forward, visto che la transazione si era chiusa con un commit prima del fallimento.

D14

Non è necessario alcun ripristino visto che è stata chiusa con un commit prima del checkpoint.

D15

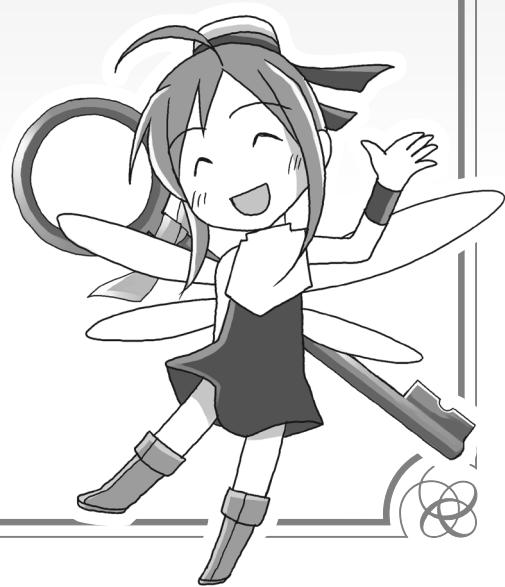
Hash.

D16

B-albero.

6

I DATABASE SONO OVUNQUE!





*GNAM SLURP!



NO, NO!

SÌ? CHE
SUCCIDE?
ANCHE TU VUOI
UNA BANANA,
RURUNA?

PADRE, DA
QUANDO SEI
TORNATO NON
HAI FATTO
ALTRO CHE
INGOZZARTI DI
FRUTTA.

SCUSA, LA
FRUTTA DI
KOD NON HA
PARAGONI!

MA DEVO AMMETTERE
CHE RURUNA HA TENUTO
BEN SALDE LE REDINI, IN
NOSTRA ASSENZA.

GUARDA COME STA
PROSPERANDO IL
REGNO DI KOD!

I DATABASE
SONO
UN'INVENZIONE
DAVVERO
UTILE!

ばばばば!!

木木木木



MA A CHI TI
RIFERISCI?
NON L'HO MAI
VISTA PRIMA
D'ORA.



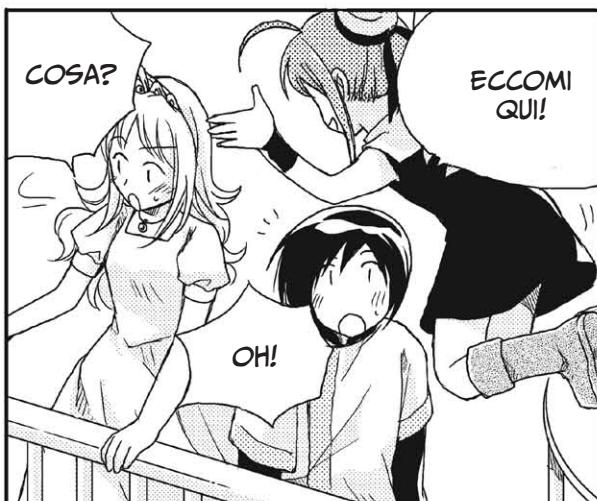
IO NO.

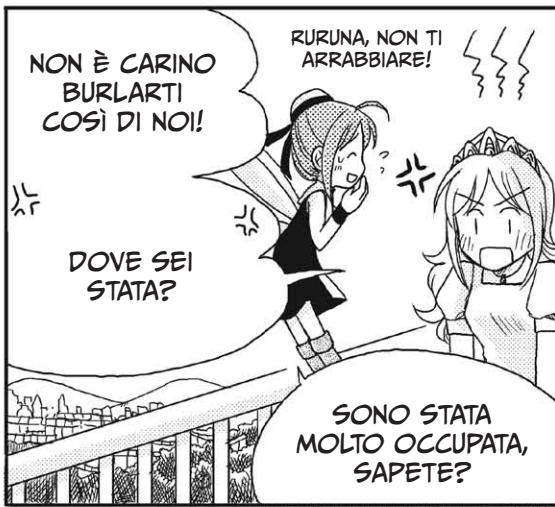
PRINCIPESSA
RURUNA.



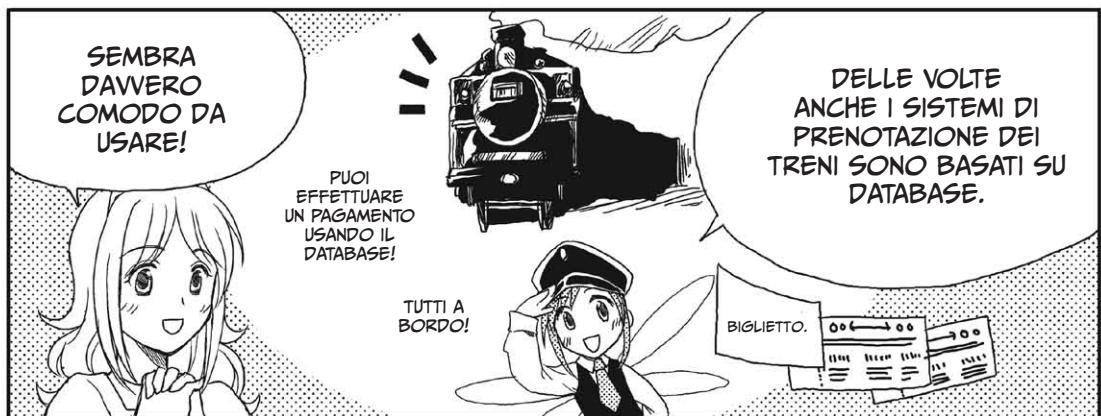


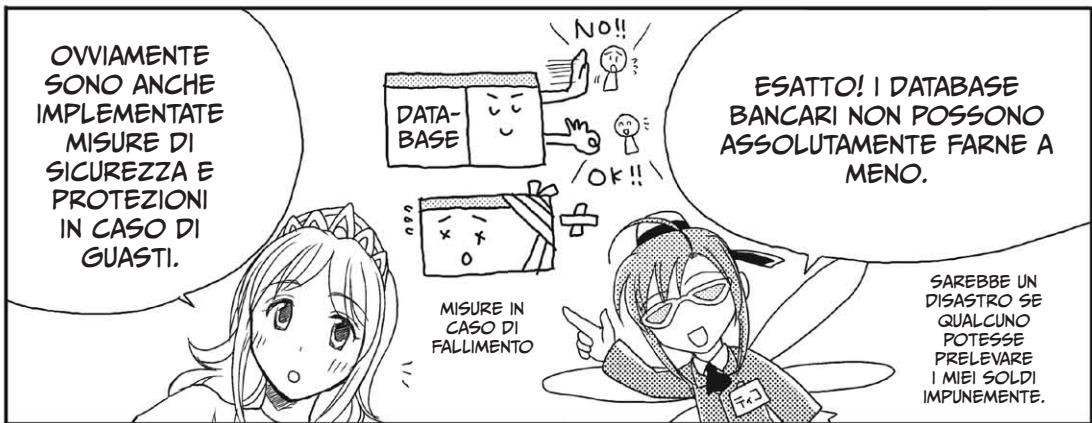
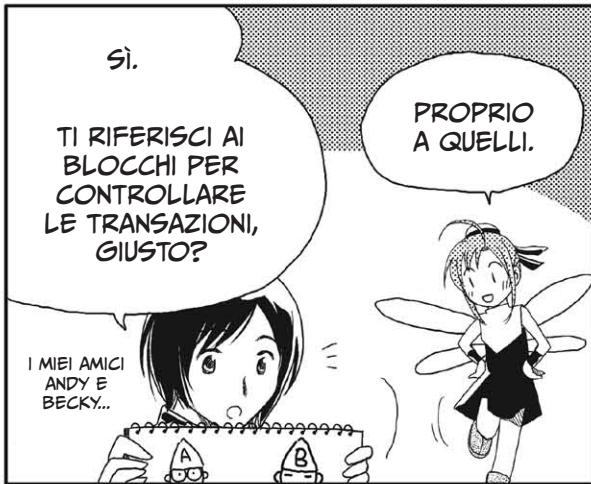
È SPARITA!



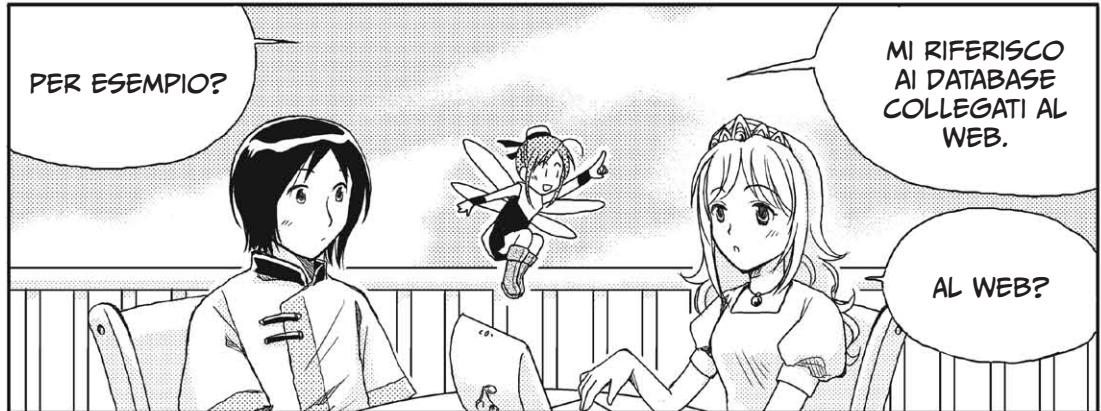


I DATABASE IN USO





I DATABASE E IL WEB



QUANDO HAI
IN MENTE
UN TIPO DI
LIBRO DA
CERCARE...

ほん、
...BASTA INSERIRE
UNA PAROLA CHIAVE
NEL BROWSER.

RICERCA
LIBRO >>>
TICO SEARCH

CATEGORIE

PAROLA CHIAVE

タイコ

TICO
SEARCH?

QUELLE TIPO
DI LIBRI TI
INTERESSA?

VEDIAMO UN
PO'... QUELLI CHE
PARLANO DI FRUTTI.

ALLORA
SCRIVI
"FRUTTI"
NEL CAMPO
PAROLA
CHIAVE.

PAROLA CHIAVE

FRUTTI

LA PAROLA CHIAVE VIENE
GESTITA COME UNA
RICHIEDA HTTP.

UN COMPUTER CHE
RICEVE ED ELABORA UNA
RICHIEDA COME QUESTA
SI CHIAMA SERVER.

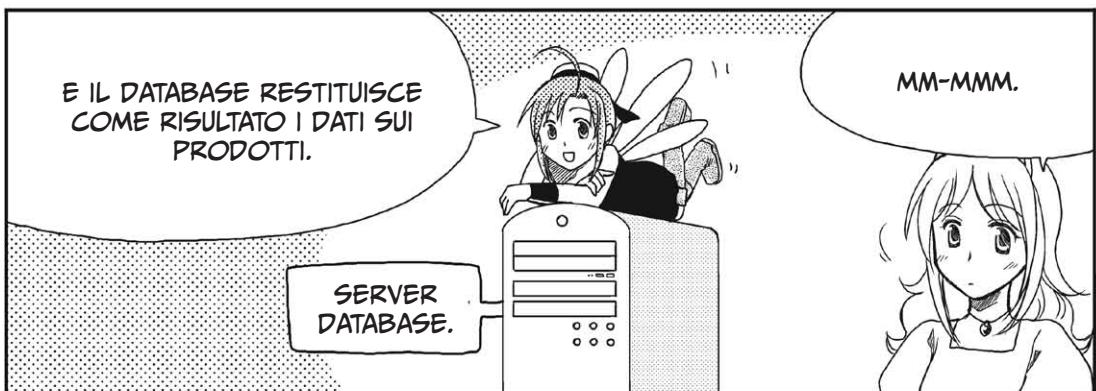
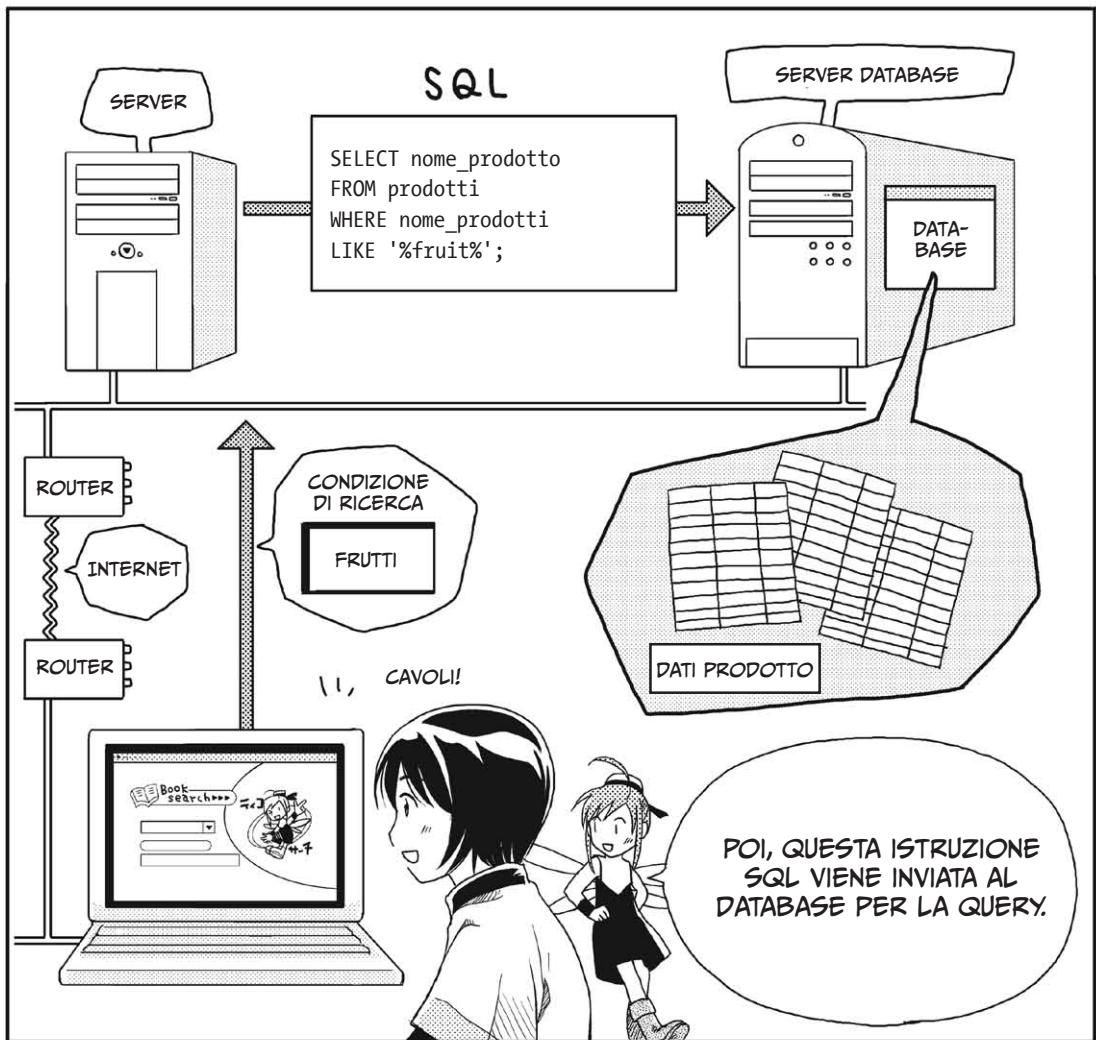
SUL SERVER
VIENE ESEGUITA
UN'ISTRUZIONE SQL,
IMMAGINO.

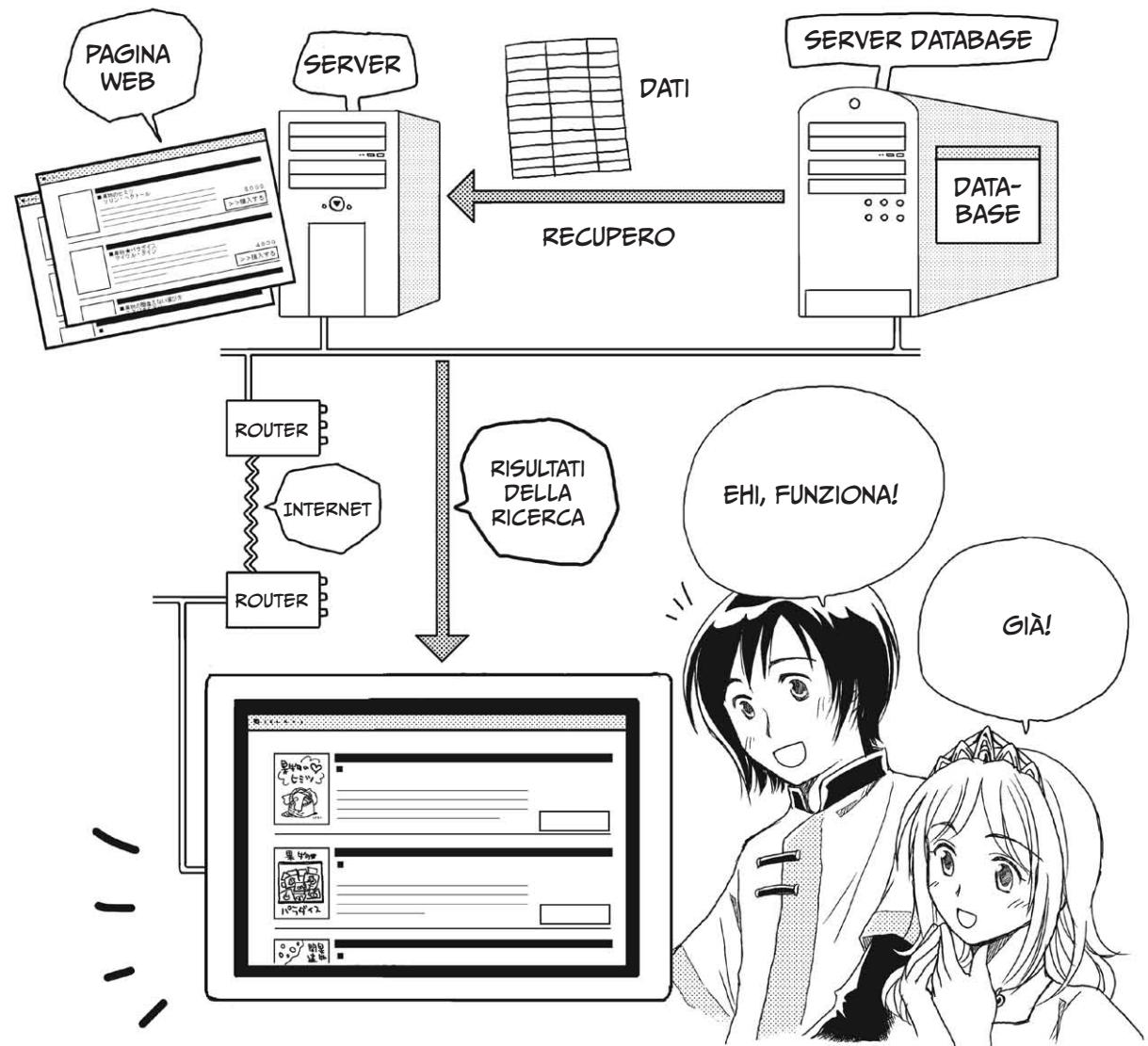
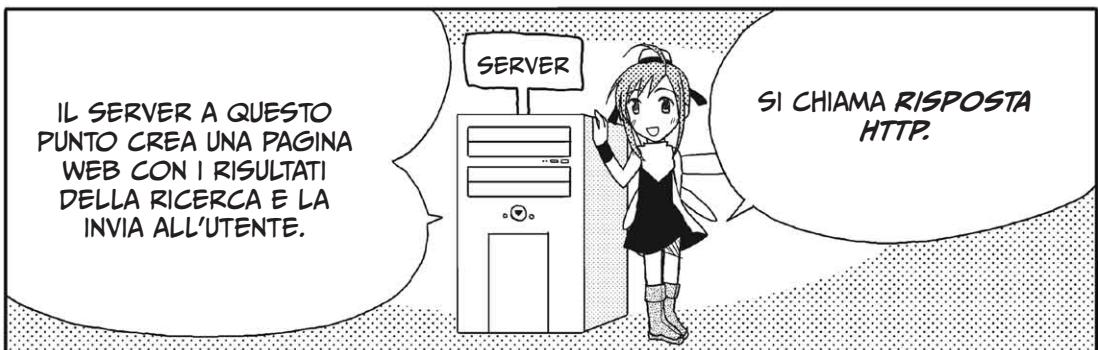
RICERCA
LIBRO >>>

CATEGORIE

PAROLA CHIAVE









IMMAGINO CHE UN SACCO DI CLIENTI ACCEDANO ALLE LIBRERIE ONLINE CONTEMPORANEAMENTE.

ANCHE CON I BLOCCHI E LE FUNZIONI DI SICUREZZA PIÙ SOFISTICATE...

...DEV'ESSERCI UN SACCO DI LAVORO PER I PROCESSORI.

IN QUEI CASI IL CARICO DI LAVORO È SUDDIVISO TRA PIÙ SERVER.

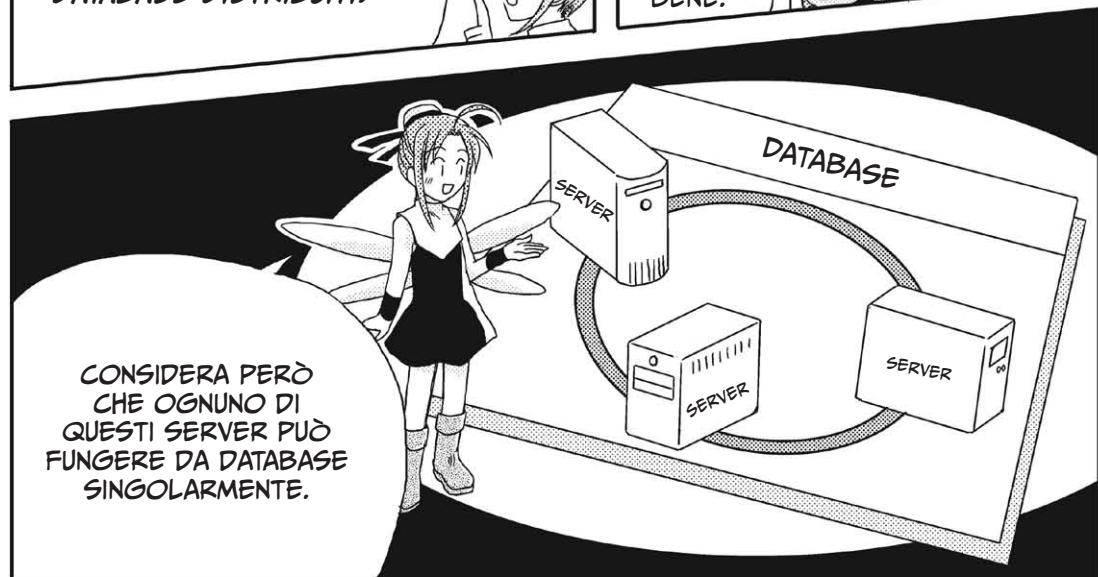
VUOI DIRE CHE NE UTILIZZANO PIÙ DI UNO?

SÌ, IL CARICO È DISTRIBUITO TRA MOLTI SERVER, PER ESEMPIO CI SARÀ UN SERVER PER IL WEB E UNO PER L'INFRASTRUTTURA (APPLICATION SERVER).

UN WEB SERVER È UN SERVER CHE CREA LA PAGINA WEB, GIUSTO?

ESATTO! MENTRE UN APPLICATION SERVER COMPONE LE ISTRUZIONI SQL, OLTRE A TANTI ALTRI COMPITI.

DATABASE DISTRIBUITI



MOLTI SERVER
VUOL DIRE ANCHE
PROTEZIONE
EXTRA CONTRO I
FALLIMENTI!!

**Knock
Out !!**

TI-CO!
TI-CO!

IN PRATICA ANCHE SE
QUALCHE SERVER È
COLPITO, IL SISTEMA
COMUNQUE RIMANE IN
PIEDI.

PUÒ INIZIARE IL
CONTEGGIO

FALLIMENTO.

WOW!

MA RICORDATE SEMPRE
CHE CI VOGLIONO ANCHE
PRECAUZIONI EXTRA PER
GESTIRE IL VOSTRO
DATABASE IN QUESTO
MODO.

PER ESEMPIO?

QUANDO UNA TRANSAZIONE
VIENE CHIUSA CON UN
COMMIT, DOVRETE
ASSICURARVI CHE VENGANO
AGGIORNATI I DATI IN TUTTO IL
DATABASE DISTRIBUITO.

POI, PER ESEMPIO,
TUTTI I SERVER DEVONO
ESSERE RIPRISTINATI
CORRETTAMENTE IN CASO DI
PROBLEMI ALLA RETE.

LE PROCEDURE ARCHIVIATE E I TRIGGER

UNA RETE È NECESSARIA SE SI VOGLIONO UTILIZZARE PIÙ SERVER.

ESATTO!
ECCO PERCHÉ VENGONO UTILI LE PROCEDURE ARCHIVIATE...

...DI SOLITO VENGONO CREATE PER ALLEGGERIRE IL CARICO DELLA RETE.

ARCHIVIATE...?

EH?

ARCHIVIATE NON VUOLE DIRE IN QUESTO CASO MEMORIZZATE?

GIUSTO!

PER RIDURRE IL CARICO SULLA RETE LE OPERAZIONI EFFETTUATE PIÙ FREQUENTEMENTE POSSONO ESSERE MEMORIZZATE IN UN DATABASE.

LE OPERAZIONI EFFETTUATE PIÙ DI FREQUENTE... MI CHIEDO QUALI POSSANO ESSERE...

BE', VISTO CHE STIAMO PARLANDO DI LIBRI, L'OPERAZIONE DI PRELIEVO DAL MAGAZZINO NELLA TABELLA INVENTARIO E L'INSERIMENTO NELLA TABELLA SPEDIZIONI...

...QUELLE NON SONO FORSE OPERAZIONI TIPICHE?

STO PENSANDO...

IN PRATICA...

...SI POSSONO MEMORIZZARE LE OPERAZIONI CHE PROBABILMENTE SARANNO SVOLTE PIÙ DI FREQUENTE...

...DIRETTAMENTE NEL DATABASE!

GRAZIE ALLA STORED PROCEDURE NON DOVREMO LANCIARE UN'ISTRUZIONE SQL...

...TUTTE LE VOLTE CHE VOGLIAMO PRELEVARE UN LIBRO DALLO STOCK E LANCIARE LA SPEDIZIONE.

CAPISCO.
IN QUESTO MODO IL CARICO DI RETE SI RIDUCE.

ANCHE IL NOSTRO LAVORO SI RIDUCE.

OH, SÌ, HAI RAGIONE!

OLTRE A QUESTE, ESISTONO ANCHE PROCEDURE CHE SI ATTIVANO AUTOMATICAMENTE.

AUTOMATICAMENTE?

QUANDO SI
AGGIORNA UN DATO,
PER ESEMPIO, UNA
PROCEDURA PUÒ
ESSERE ESEGUITA
AUTOMATICAMENTE.

IN QUESTO CASO
PARLIAMO DI
TRIGGER.

Trigger,
come...

...il
grilletto!

FUNZIONA COME IL
GRILLETTO DI UNA
PISTOLA!

PREMI IL GRILLETTO E
PARTE UN PROGETTILE.
AGGIORNA UN DATO E VIENE
ESEGUITA UNA PROCEDURA.

SAREBBE DAVVERO
PRATICO SE TUTTE
LE VOLTE CHE
ARRIVA UN ORDINE
E IL DATABASE SI
AGGIORNA...

PERCHÉ SONO
VESTITO ANCH'IO
COSÌ?

...PARTISSE IN
AUTOMATICO
UNA PROCEDURA
CHE RIDUCE
L'INVENTARIO
E PREPARA LA
SPEDIZIONE.

IL SEMPLICE ACQUISTO
DI UN LIBRO GENERA
UN SACCO DI LAVORO
DIETRO LE QUINTE,
VERO?



ESATTO.

VERO.

ANCHE SE, NELLA MAGGIOR PARTE DEI CASI, IL DATABASE NON SI Vede...

...QUANDO ACQUISTI UN LIBRO IN RETE.

CLACK

QUANTO A NOI, DOVREMO IMPARARE TUTTE QUESTE COSE, PASSO DOPO PASSO.

CHE NE DICI, CAIN?

SONO D'ACCORDO, PRINCIPESSA.

TICO CI HA INSEGNATO COME SI USANO I DATABASE, ORA POSSIAMO CONTINUARE IN AUTONOMIA.

BENE!

NON DIMENTICATE MAI CHE I DATABASE SONO IL MODO MIGLIORE DI ORGANIZZARE I DATI IN MODO EFFICIENTE.

MI PIACE! SONO PRONTA, USANDO I DATABASE...

...COSTRUIRÒ UN REGNO MERAVIGLIOSO DOVE TUTTI POTRANNO GODERE DI UNA VITA CONFORTEVOLe.

FAREMO DEL NOSTRO MEGLIO!

!!

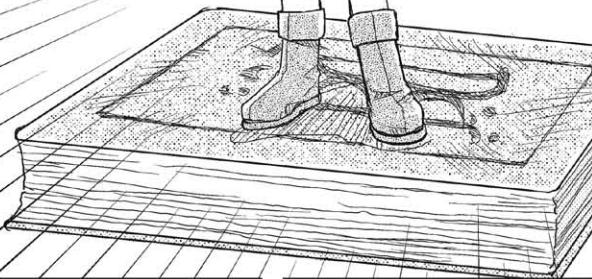
!!

SI-i-i-i!



LA FATINA
DEI
DATABASE!

びし、



OGGI ERO VENUTA
SOLO A DIRVI ADDIO.

MA NONOSTAN-
TE LE INTENZIONI, MI
SONO TRATTENUTA
DI PIÙ.

È STATO BELLO
CONOSCERVI, ANCHE
SE È DURATO POCO!



TICO...

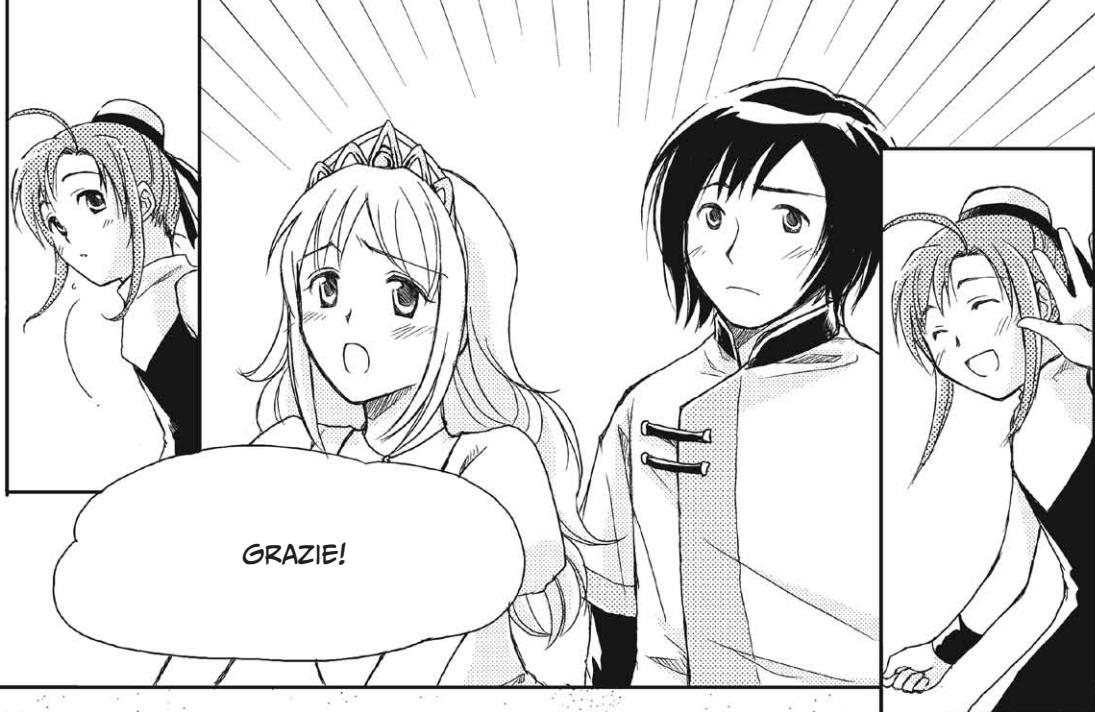
TICO, ASPETTA!

DEVO...

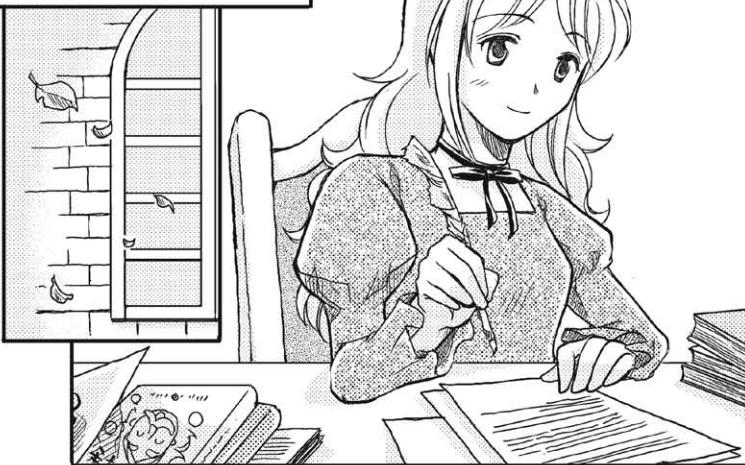
MIA CARA
TICO!



TICO!!



IL TEMPO PASSA...



PRINCIPESSA, VA
TUTTO BENE CON
QUEL SUO LIBRO SUI
DATABASE?



SÌ!

VUOI DARE
UN'OCCHIATA?

CERTO!

ORA LO STO
SEMPLIFICANDO
PER FARE IN MODO
CHE TUTTI POSSANO
CAPIRE.



FARLO A
FUMETTI È
STATA DAVVERO
UN'OTTIMA
IDEA.

E CAIN È UN
DISEGNATORE
ECCELLENTE.

CHE DISEGNI CARINI... ❤



GUARDA...

...QUESTA SARÀ LA
COPERTINA.

SPLENDIDA!





UN GIORNO, DA UN
ANTICO LIBRO SUI
DATABASE...

I DATABASE NEL
REGNO DI KOD



...SBUCÒ
FUORI UNA
PICCOLA
FATINA...

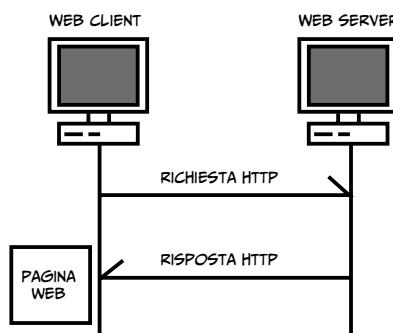
FINE

I DATABASE NEL WEB

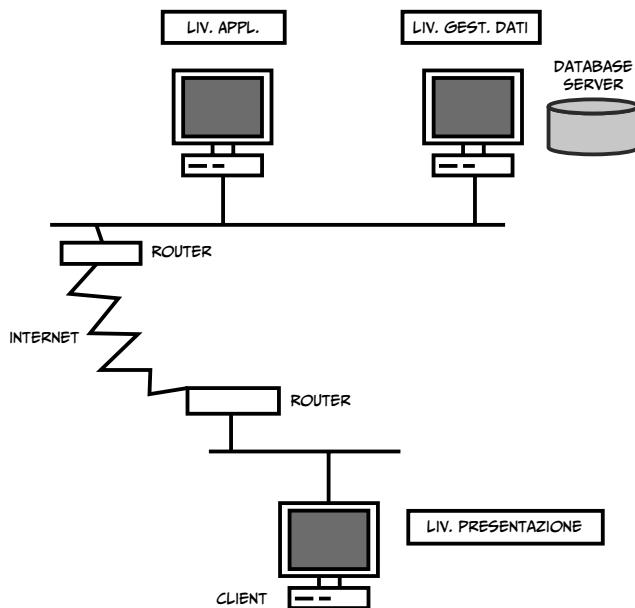


I database vengono usati per molti scopi, per esempio nei sistemi di prenotazione ferroviaria e nei circuiti bancari. Sono indispensabili nella vita di tutti i giorni e nelle operazioni economiche. Come ho mostrato a Ruruna e a Cain, i database su server sono molto diffusi. In un sistema basato su internet, il protocollo di comunicazione che si usa è l'HTTP (HyperText Transfer Protocol). Il server attende una richiesta da un utente, e quando la riceve (si chiama richiesta HTTP) risponde via software tramite una pagina web generata appositamente (risposta HTTP).

Una *pagina web* consiste di vari testi in formato HTML. Altri file specificati da URL (Uniform Resource Locator) possono comparire all'interno di una pagina web per fornire informazioni aggiuntive (per esempio immagini).



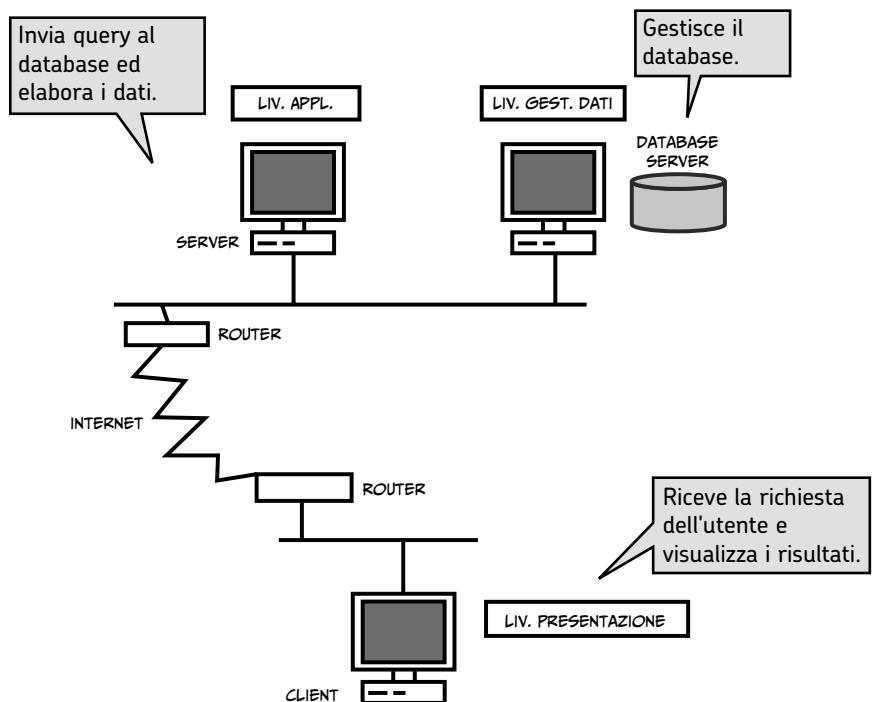
Quando un database è usato in associazione a una pagina web, allo schema precedente si aggiunge un server database. Il sistema può essere configurato in tre livelli, e prende il nome di *architettura client-server a tre strati (three-tier)*. Un sistema di questo tipo comprende tre livelli (di presentazione, di applicazione e di gestione dei dati).



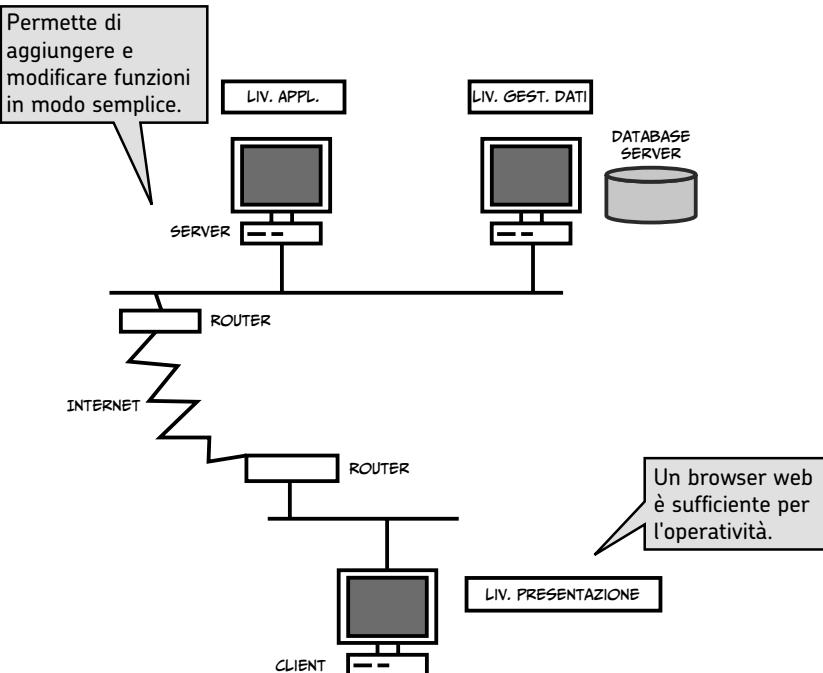
Il *livello di presentazione* riceve una richiesta dall'utente, ad esempio una condizione di ricerca, che dev'essere condivisa con il database. Questo livello serve anche a elaborare i risultati ricevuti dal database perché siano visualizzati sullo schermo dell'utente. Un browser web (come Chrome o Firefox) funziona come strumento di presentazione e interfaccia per l'utente.

Il *livello di applicazione* si occupa di elaborare i dati, ed è qui che vengono composte le istruzioni SQL. I processi così gestiti possono essere scritti utilizzando linguaggi di programmazione diversi. A seconda dei contenuti e della quantità di processi, potrebbe essere necessario utilizzare più server (ad esempio un application server e un web server) per gestire tutti i processi.

Il *livello di gestione dei dati* elabora i dati su un database server. I risultati della ricerca sono restituiti dal database in risposta a una query SQL.



L'architettura client-server a tre strati è un sistema semplice e flessibile. Per esempio, quando si implementano nuove aggiunte o modifiche, permette di separare la porzione che ci interessa modificare come parte del livello di applicazione. Nel livello di presentazione, poi, permette di usare un browser web eliminando la necessità di installare software specifici.



USARE LE PROCEDURE ARCHIVIATE

In un sistema basato sul web, l'incremento del traffico in rete può generare dei problemi. Per fortuna puoi memorizzare all'interno del database server delle procedure per alleggerire lo scambio dei dati.

Memorizzare procedure nel database server consente di ridurre il carico sulla rete perché elimina il bisogno di trasferire di frequente query SQL. Tra l'altro, le procedure semplificano lo sviluppo delle applicazioni, visto che processi standard possono essere incapsulati in applicazioni pronte all'uso. In realtà, le procedure non sono che un aspetto del mondo molto più vasto dei *programmi memorizzati*. Le altre tipologie di programma sono le *funzioni* e i *trigger*.

TIPOLOGIE DEI PROGRAMMI MEMORIZZATI

Tipologia	Definizione
Procedura	Programma che non restituisce valori dalla procedura in corso
Funzione	Programma che restituisce valori dalla procedura in corso
Trigger	Programma lanciato automaticamente prima o dopo le operazioni del database



DOMANDE

Riesci a rispondere a queste domande? Troverai le risposte a pagina 205.

D1

In un'architettura client/server a tre strati, in quale dei tre livelli opera il database?

In un'architettura client/server a tre strati, quale dei tre livelli interagisce con l'utente e mostra i risultati?

CHE COS'È UN DATABASE DISTRIBUITO?



In un sistema pensato per il web, l'operatività è distribuita tra un database server, un web server e un browser web, ognuno dei quali risponde a esigenze specifiche. Questo genere di sistema distribuito consente maggiore flessibilità e diminuisce le esigenze in termini di capacità di calcolo dei singoli server.

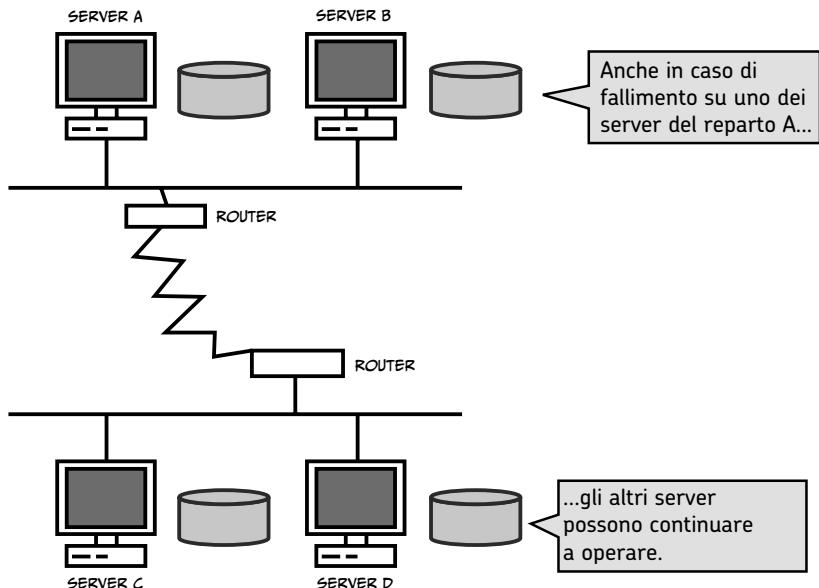
Ma anche un database server può essere distribuito su più server separati. I database server distribuiti possono trovarsi in luoghi diversi o sulla stessa rete. C'è però da sottolineare che un database distribuito può essere gestito come se fosse un singolo. Se il database distribuito appare come se fosse caricato su un solo server, l'utente non deve preoccuparsi dell'ubicazione dei dati o del loro trasferimento.

Un database, come vedremo, può essere distribuito orizzontalmente o verticalmente.

LA DISTRIBUZIONE ORIZZONTALE

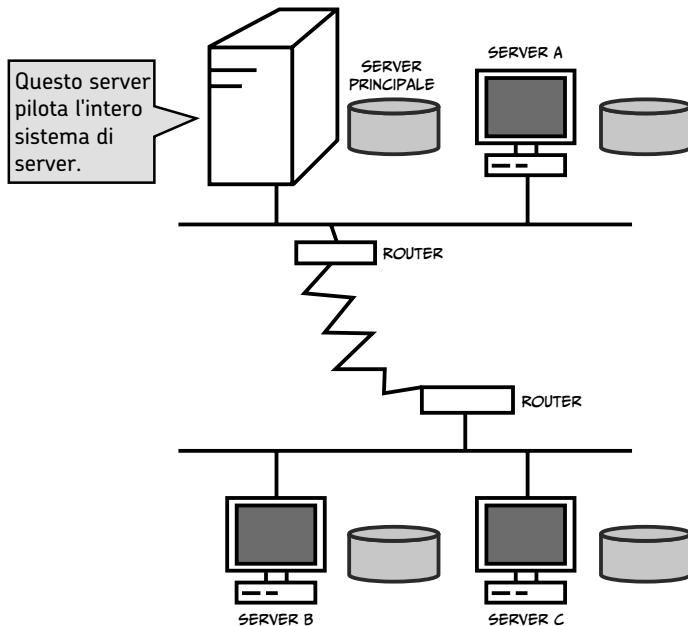
La *distribuzione orizzontale* utilizza diversi database server paritari. Ognuno di loro può elaborare dati provenienti dagli altri e, parametri, ognuno di loro si rende disponibile agli altri. Questa struttura viene usata per sistemi di database che operano separatamente in ogni reparto.

Un database distribuito orizzontalmente è un sistema progettato per essere particolarmente robusto, visto che un fallimento su uno dei server non inficia l'operatività del database.



LA DISTRIBUZIONE VERTICALE

La *distribuzione verticale* assegna ai server compiti diversi. Uno dei server funzionerà come server principale (*main server*) e avrà dunque un ruolo chiave, mentre gli altri server riceveranno compiti più specifici. Una distribuzione di tipo verticale semplifica la gestione del server principale, anche se lo "condanna" a gestire carichi maggiori. Un esempio di distribuzione verticale potrebbe essere il server principale di una grande azienda, a cui sono collegati i server relativi a ogni singolo reparto.



PARTIZIONAMENTO DEI DATI

In un database distribuito i dati vengono memorizzati su server diversi. È molto importante stabilire correttamente come suddividere i dati. I dati possono essere divisi nei seguenti modi.

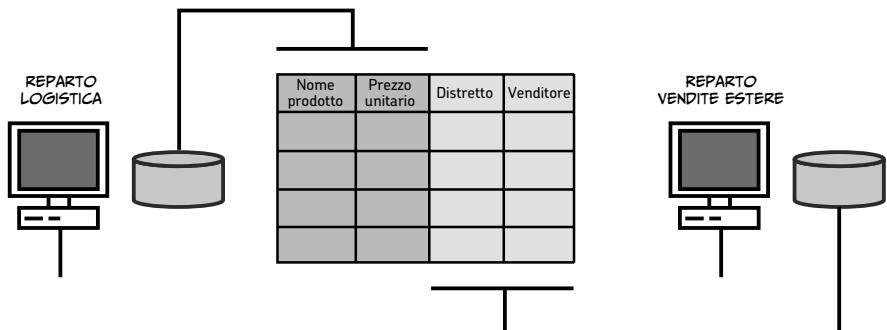
IL PARTIZIONAMENTO ORIZZONTALE

Un *partizionamento orizzontale* divide i dati separandoli per righe. Le righe che risultano da questa divisione vengono poi distribuite su più server. Questa forma di partizionamento è spesso usata quando i dati possono essere suddivisi in modo da poter raggruppare sullo stesso server dati collegati tra loro (e, per questo, oggetto spesso di accessi simultanei).



IL PARTIZIONAMENTO VERTICALE

Un *partizionamento verticale* divide i dati per colonne. Le colonne che risultano da queste divisioni vengono poi distribuite su vari server. Per esempio, un partizionamento verticale è consigliato per gestire e unire database indipendenti che provengono da reparti diversi, come il Reparto Logistica, il Reparto Vendite e il Reparto Esportazione.



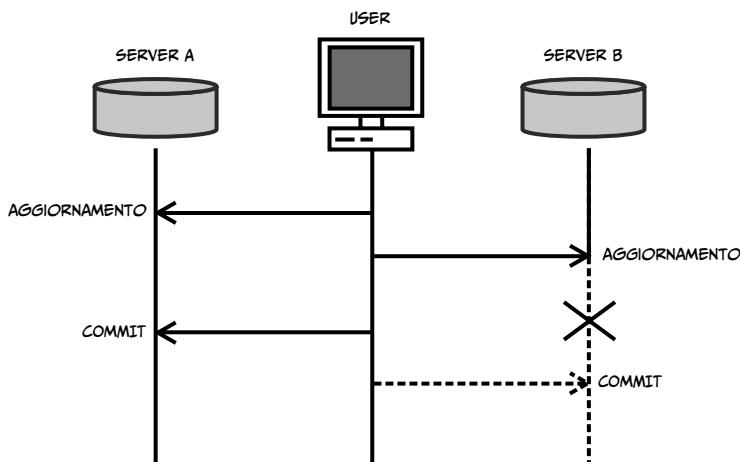
PREVENIRE LE INCOERENZE CON UN COMMIT A DUE FASI



I database ospitati da server diversi in un database distribuito possono essere configurati per comportarsi come un database singolo agli occhi degli utenti. Sono però necessari diversi passaggi per superare il fatto che i dati sono effettivamente memorizzati all'interno di server separati.

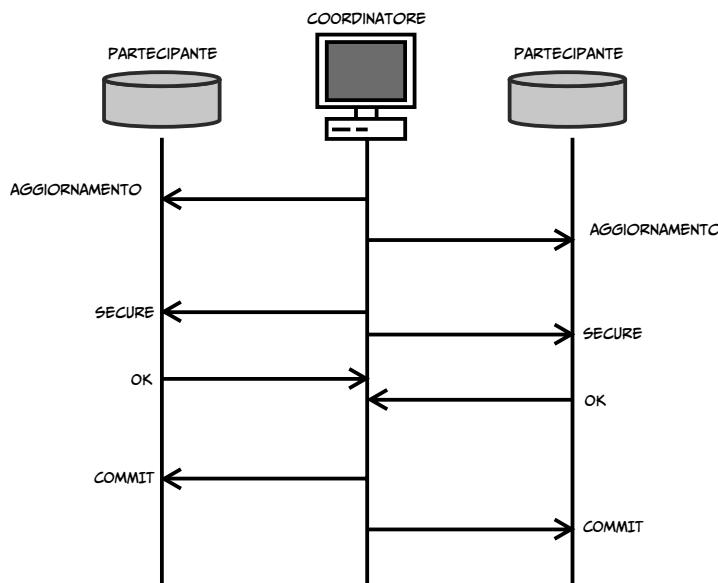
Per prima cosa, ogni volta che viene eseguita un'operazione su un dato, tutti i dati su tutti i server devono essere aggiornati di conseguenza.

In un database distribuito, il "commit" standard potrebbe condurre ad asincronie negli aggiornamenti, come mostrato qui sotto. È una violazione del principio di atomicità delle transazioni, visto che questa transazione è destinata a non concludersi né con un commit né con un rollback, e questo renderebbe incoerente l'intero sistema database.

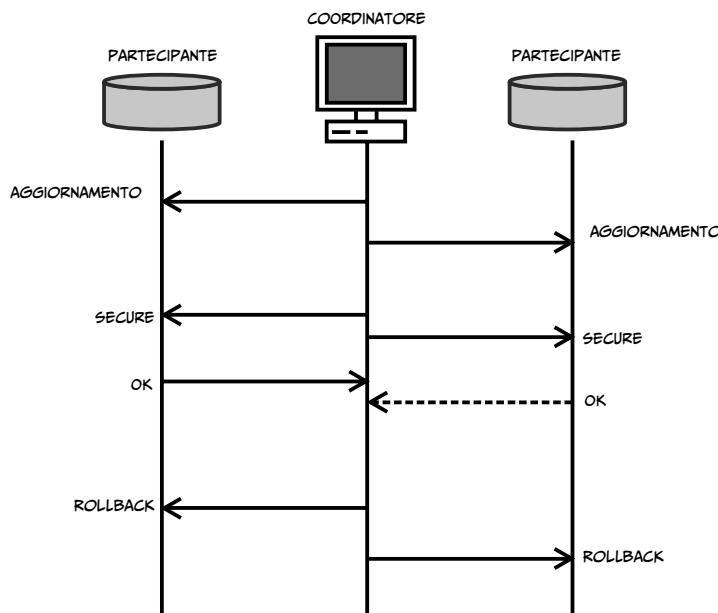


Quindi, un *commit a due fasi* viene adottato nei database distribuiti. Il commit a due fasi crea un'operazione di commit sia dalla prima che dalla seconda operazione di commit.

Un commit a due fasi prevede la presenza di un coordinatore e di vari partecipanti. All'inizio del commit a due fasi, il coordinatore chiede ai partecipanti se un commit è possibile. In questo caso, i partecipanti rispondono OK. Questa fase preliminare è nota con il nome di *preparazione* (*prepare*). Nella seconda fase, il coordinatore invia le istruzioni per il commit, e tutti i partecipanti lo eseguono di conseguenza.



Se uno dei nodi non può assicurare l'operazione nel commit a due fasi, tutti i partecipanti ricevono l'istruzione di eseguire un rollback. Così i database su tutti i server restano consistenti fra di loro.





DOMANDE

Prova a rispondere a queste domande sui commit a due fasi. Troverai le risposte a pagina 205.

D3

In uno schema che prevede commit a due fasi, quali istruzioni invia il coordinatore durante la prima fase?

D4

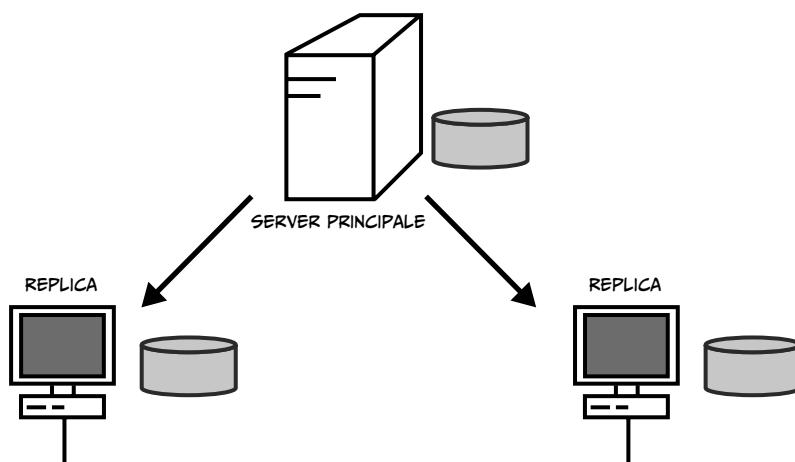
In uno schema che prevede commit a due fasi, quali istruzioni invia il coordinatore durante la seconda fase?

LA REPLICAZIONE DI UN DATABASE

Alcuni database distribuiti prevedono duplicati, o repliche, che servono a ridurre il carico sulla rete. Questa pratica prende il nome di *replicazione*. Il database principale viene definito *master*, mentre la copia è definita *replica*. Esistono molti tipi diversi di replicazione.

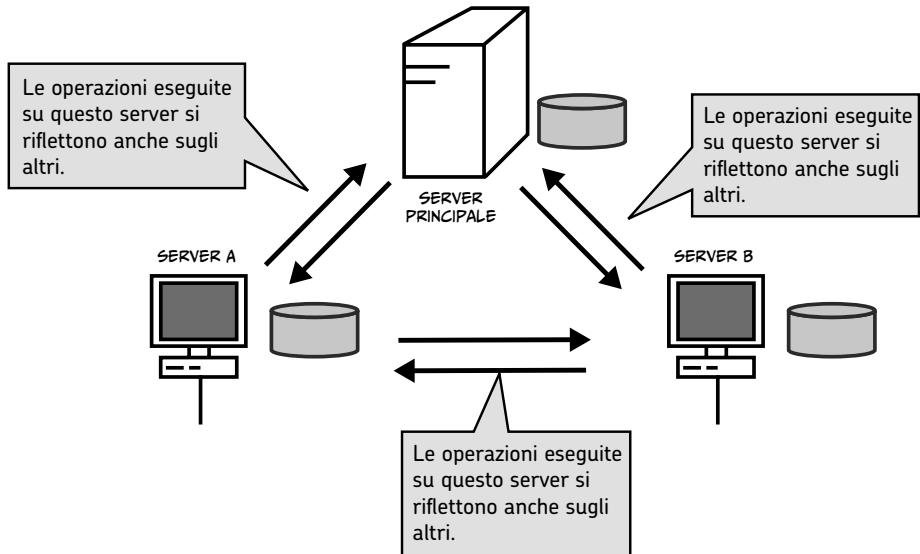
LA MODALITÀ DI SOLA LETTURA

Una replica di sola lettura viene creata e copiata dal master e resa disponibile. Per modificare i dati l'utente deve essere connesso al server principale.



LA REPLICAZIONE CON ACCESSO A TUTTI I SERVER

In questo metodo, il master è condiviso da tutti i server. Gli aggiornamenti eseguiti su un server si riflettono anche sugli altri.



ULTERIORI APPLICAZIONI DEI DATABASE



Quest'ultima sezione introduce tecnologie applicate che si basano sempre sui database.

XML

L'XML (*eXtensible Markup Language*) sta diventando sempre più popolare come metodo di conservazione dei dati. L'XML rappresenta i dati personalizzandoli con dei *tag*. Visto che questi tag forniscono a loro volta informazioni sui dati a cui sono riferiti, questo linguaggio è molto adatto alla conservazione e alla ricerca dei dati.

L'XML è utile perché la sua grammatica rigidamente strutturata semplifica i processi di programmazione. Inoltre, l'XML viaggia su file di testo (facilmente editabili) e può comunicare con altri sistemi. Per queste ragioni, l'XML è talvolta usato come metodo per la rappresentazione dei dati in alternativa al database.

```
<?xml version="1.0"?>
<prodotti>
  <frutto>
    <codice prodotto>101</codice prodotto>
    <nome prodotto>Melone</nome prodotto>
    <prezzo unitario>800</prezzo unitario>
  </frutto>
  <frutto>
    <codice prodotto>102</codice prodotto>
    <nome prodotto>Fragola</nome prodotto>
    <prezzo unitario>150</prezzo unitario>
  </frutto>
  <frutto>
    <codice prodotto>103</codice prodotto>
    <nome prodotto>Mela</nome prodotto>
    <prezzo unitario>120</prezzo unitario>
  </frutto>
</products>
```

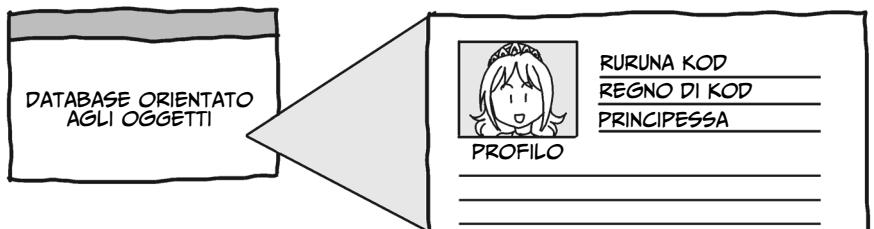
I DATABASE ORIENTATI AGLI OGGETTI

Un database relazionale memorizza i dati in una tabella. Tuttavia, un database relazionale potrebbe rivelarsi poco adatto a gestire determinati tipi di dato. Ecco dove entrano in gioco i database orientati agli oggetti (OODB, *Object-Oriented DataBase*).

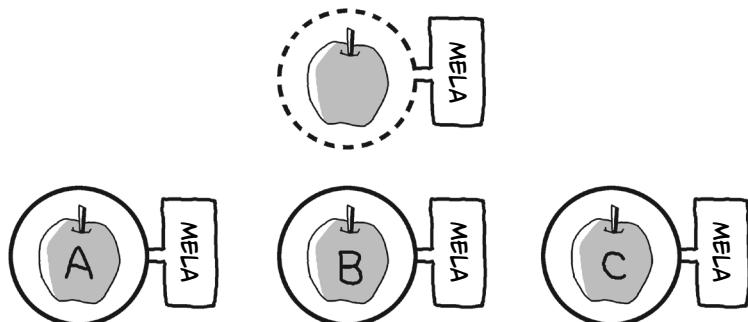
Questa metodologia utilizza *oggetti*, ovvero insiemi di dati e di istruzioni su come quei dati dovrebbero essere utilizzati. Puoi per esempio nascondere i dati e rendere visibili solo le operazioni sui dati, e quindi gestire l'oggetto come un componente indipendente. Questa tecnica viene anche chiamata *incapsulamento*.

In un database orientato agli oggetti, ogni oggetto è rappresentato da un *identificatore*. A volte, un oggetto può essere anche chiamato *istanza*.

In un OODB puoi anche gestire oggetti composti, vale a dire oggetti che ne contengono altri al loro interno. Questo, per esempio, significa che puoi memorizzare un'immagine combinata a un testo come se fossero un unico oggetto. Il database orientato agli oggetti consente quindi una gestione più flessibile di dati composti.

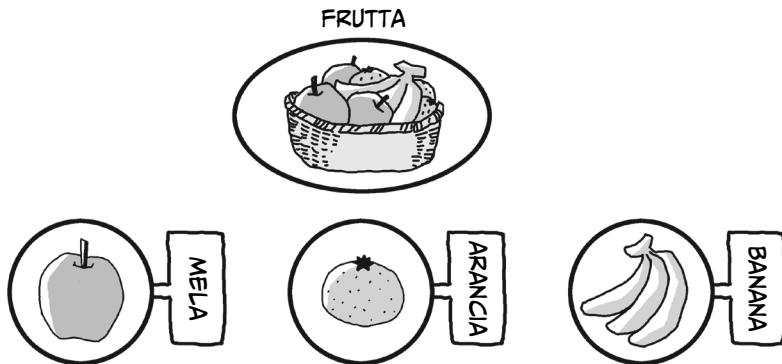


In un database orientato agli oggetti, diversi concetti possono semplificare lo sviluppo. Il modello di gestione degli oggetti è chiamato *classe*. Per esempio, immagina di avere progettato una classe Mela. Gli oggetti (o istanze) in quella classe potrebbero essere Mela A, Mela B e così via. La classe Mela consente la creazione di questi oggetti.



In uno schema orientato agli oggetti, una classe può anche essere inserita in una gerarchia. Puoi creare una classe "figlia" che conterrà gli stessi dati e le stesse funzioni della classe "madre". Questa relazione prende il nome di *ereditarietà*. Puoi anche assegnare funzioni specifiche alla classe figlia.

Per esempio, la classe Mela e la classe Arancia possono ereditare i dati e le funzioni della classe Frutta e possedere al contempo dati e funzioni specifiche. In uno schema orientato agli oggetti le relazioni gerarchiche servono a garantire maggiore efficienza.



Riassumendo



- L'architettura client/server a tre strati è un metodo di configurazione per sistemi basati sul web.
- Un database funge da livello di gestione dei dati.
- Un sistema database distribuito gestisce database allocati su risorse separate.
- Il commit a due fasi è un metodo usato nei database distribuiti.

Risposte

- D1** Il livello di gestione dei dati.
- D2** Il livello di presentazione.
- D3** Preparazione (*prepare*).
- D4** Commit o rollback.

Conclusioni

Ti è piaciuto studiare i database? Dovrai imparare ancora molte cose prima di essere in grado di gestire tutti gli aspetti di un database, ma i principi fondamentali rimarranno sempre gli stessi. Se capirai bene le regole di base, riuscirai a identificare i dati significativi nel mondo reale, che è il primo passo per progettare e utilizzare un database. Poi, lavorando sulle conoscenze acquisite, potrai completare la tua competenza con tantissime tecniche avanzate. Buona fortuna!

LE ISTRUZIONI SQL PIÙ USATE

QUERY DI BASE

```
SELECT nome_colonna, ...
FROM nome_tabella;
```

QUERY CONDIZIONALE

```
SELECT nome_colonna, ...
FROM nome_tabella
WHERE condizione;
```

PATTERN MATCHING

```
SELECT nome_colonna, ...
FROM nome_tabella
WHERE nome_colonna LIKE 'pattern';
```

RICERCA ORDINATA

```
SELECT nome_colonna, ...
FROM nome_tabella
WHERE condizione
ORDER BY nome_colonna;
```

AGGREGAZIONE E RAGGRUPPAMENTO

```
SELECT nome_colonna, ...
FROM nome_tabella
WHERE condizione
GROUP BY nome_colonna_per_raggruppamento
HAVING condizioni_per_righe_raggruppate
```

UNIONE DI TABELLE

```
SELECT nome_tabella1.nome_colonna, ...
FROM nome_tabella1, nome_tabella2, ...
WHERE nome_tabella1.nome_colonna = nome_tabella2.nome_colonna
```

CREAZIONE DI UNA TABELLA

```
CREATE TABLE nome_tabella(  
nome_colonna1 tipo_di_dato,  
nome_colonna2 tipo_di_dato,  
...  
);
```

CREAZIONE DI UNA VISTA

```
CREATE VIEW nome_vista  
AS SELECT istruzione
```

CANCELLAZIONE DI UNA TABELLA

```
DROP TABLE nome_tabella;
```

CANCELLAZIONE DI UNA VISTA

```
DROP VIEW nome_vista;
```

INSERIRE UNA RIGA

```
INSERT INTO nome_tabella(nome_colonna1, ...)  
VALUES (valore1, ...)
```

MODIFICARE UNA RIGA

```
UPDATE nome_tabella  
SET nome_colonna = valore1, ...  
WHERE condizione;
```

CANCELLARE UNA RIGA

```
DELETE FROM nome_tabella  
WHERE condizione;
```

INDICE

A

accesso, 19, 106, 126–129, 141–142, 159–160, 167
aggregazione, funzione di, 98–100, 110–111
ALL (istruzione), 159
application server, 182, 195
architettura client-server a tre strati (three-tier), 194–196, 197, 205
atomicità, 153–154
AVG, media (funzione), 98, 99, 110

B

B-albero, indicizzazione, 163
backup, copie, 161
base, tabelle,
(v. tabelle, base)
blocco,
- a due fasi, 156–157
- condiviso, 133, 155–156
- controllo, 131–137, 155–157, 167, 175–176, 182
- esclusivo, 134–136, 155–156
- granularità, 157
buffer, 161

C

campo, 27–28, 30, 34, 35, 48
caratteri,
- jolly, 97, 108
- stringa di, 84, 108
cardinalità, 74
checkpoint, 161–162
chiave,
- esterna, 44, 48, 72, 101
- primaria, 35, 44, 48, 65, 67, 72, 78–79, 101, 103, 115
cicli annidati, metodo, 165
coerenza, 153, 154–155, 184
colonne, 34, 84
COMMIT (istruzione), 133, 137, 150, 154, 205
commit a due fasi,
199–201, 205
conflitto di dati, (v. dati, conflitto di)
confronto, operatori di,
(v. operatori di confronto)
congiunzione (operazione), 37, 43, 44, 48, 165
congiunzione di tabelle,
(v. tabelle, congiunzione/ unione)

controlli di concorrenza,
- blocco, 131–137, 155–157, 167
- livelli di isolamento, 158
- ottimistico, 158
- sull'orario di accesso, 158
controllo,
- accesso utenti, 19, 106, 126–129, 141–142, 159–160, 167
- ottimistico, 158
- sull'orario di accesso, 158
corruzione di dati, (v. dati, corruzione di)
COUNT (funzione), 99–100, 110
CREATE TABLE (istruzione), 103, 115–119
CREATE VIEW (istruzione), 117

D

database, 6, 10, 15, 187
- efficienza,
3–4, 15, 19, 146, 174
- fallimento, 161, 184, 197
- incoerente, 153, 154, 159, 199–201
- inserimento su sistemi esistenti, 14
- master, 201–202
- modello, 33, 39
- modello gerarchico, 32, 33, 39, 204
- note, 30–31
- replicazione, 201–202
- terminologia, 24–31
- tipi, 32–39
- uso, 19–21, 175–182
database distribuiti, 183–184, 197–199, 205
- commit a due fasi, 199–201, 205
- distribuzione orizzontale, 197
- distribuzione verticale, 198
- partizionamento dei dati, 198–199
- replicazione, 201–202
database, progettazione, 19, 26, 74, 81, 84
- E-R (Entità-Relazione), modello, 50–55, 74–77, 81
- normalizzazione, 60–72, 78–81
dati,
- conflitto, 13, 17–18, 21, 60, 71, 116, 153, 158
- corruzione, 20, 154
- duplicati, 11, 16, 18, 19, 21, 29
- elaborazione, 35–37, 47–48, 130, 159, 167, 182, 195–198
- estrazione, 36–37, 39–47
- gestione, 10, 19
- inserimento, 21, 90–92, 103–104, 106, 116
- organizzazione, 32–39
- perdita, 20, 154
- protezione, 19, 127, 138–142, 159–160, 161–164, 167, 176, 182, 184
- recupero, 20, 36–47, 90–92, 95–99, 101–102, 106, 147–152, 161–164, 167, 180, 202
- ricerca, 93–97, 106, 108, 112–115, 163.
(v. anche SQL)
- sicurezza, 19, 127, 138–142, 159–160, 161–164, 167, 176, 182, 184
dati condivisi, problemi con, 12, 20, 21, 129, 175
DBMS (DataBase Management System), 21
DCL (Data Control Language), 106
DDL (Data Definition Language), 106
deadlock, 136, 158
DELETE (istruzione), 104, 116, 118, 119, 159
differenza (operazione), 37, 39, 41
disastro, in caso di, 20, 147–150, 161–164, 167
distribuzione
- orizzontale, 197
- verticale, 198
divisione, operazione, 37, 43, 45
divisione tabelle (v. normalizzazione)
DML (Data Manipulation Language), 106
DROP TABLE/DROP VIEW (istruzione), 118
durabilità, 153, 159–160

E

elaborazione dati,
(v. dati, elaborazione)
entità, 52–54, 74

- E-R (Entità-Relazione), modello, 50–55, 74–77, 81**
- estrazione dati, (v. dati, estrazione)**
- F**
- fallimento fisico (media failure), 161**
- fallimento, tipologie, 161
- forma, 62–70, 81–82
- non normalizzata, 62, 78–79
- forma normale,
- prima, 62–64, 66, 78–79
 - seconda, 62, 64, 66–69, 78–79, 82
 - terza, 62, 68, 69–70, 78–79, 81, 82
- funzioni archiviate, 196
- G**
- gestione dati
- scoordinata, 10
 - unificata, 19
- gestione separata, 11, 19, 72
- GRANT (istruzione), 159, 168**
- granularità, 157
- dei blocchi, 157
 - fine, 157
- GROUP BY (frase), 110**
- H**
- hash (funzione), 167
- hash, indicizzazione, 163
- HAVING (frase), 111**
- HTML (HyperText Markup Language), 194**
- HTTP (HyperText Transfer Protocol), 178, 180, 194**
- I**
- incapsulamento, 203
- indice/indicizzazione, 143–147, 162–164, 167
- inner join, 115
- inserimento dati, (v. dati, inserimento)
- INSERT (istruzione), 104, 116, 119, 159**
- Internet database. (v. Web, database sul)
- intersezione, operazione, 37, 39, 41
- ISBN (International Standard Book Number), 45**
- ISO (International Organization for Standardization), 124**
- isolamento, 153, 155–158
- livelli di, 158
- istanza, 203
- L**
- lettura,
- fantasma, 158
 - non ripetibile, 158
 - sporca, 158
- lettura, operazione, 130, 133, 134, 159
- LIKE (istruzione), 97, 108**
- livello, 194–196, 205
- applicativo, 194–196
 - di presentazione, 205
- log, 148–149
- M**
- master database, (v. database, master)
- MAX (maximum value) (funzione), 99–100, 110**
- meccanismi di ripristino, (v. ripristino, meccanismi)
- media failure, (v. fallimento fisico)
- memoria, (v. procedure archiviate)
- MIN (minimum value) (funzione), 99, 110**
- modelli di organizzazione dati, (v. dati, organizzazione)
- modello relazionale, 33–34, 35, 39, 47, 48
- molti-a-molti, relazione, 55, 74, 75, 81
- N**
- nome utente, (v. utente, nome)
- normalizzate, tabelle, 72, 91
- normalizzazione, 60–72, 78–81
- note, 30–31
- null, 30–31, 108
- O**
- oggetti composti, 203
- OODB (Object-Oriented DataBases), 203–205**
- operatore, 107
- di confronto, 107
 - di insieme, 39–42
 - logico, 107
 - relazionale, 43–47
- operazione,
- congiunzione, 37, 43, 44, 48, 165
 - differenza, 37, 39, 41
 - divisione, 37, 43, 45
 - estrazione dati, 39–47
 - intersezione, 37, 39, 41
 - prodotto cartesiano, 37, 39, 42
- proiezione, 36, 37, 43, 165
- scrittura, (v. scrittura, operazioni)
- selezione, 37, 39, 43, 47, 48, 165
- unione, 37, 39, 40, 48
- ORDER BY (istruzione), 98**
- ordinare. (v. aggregazione, funzione di)
- orizzontale,
- distribuzione, (v. distribuzione orizzontale)
 - partizionamento, (v. partizionamento orizzontale)
- ottimizzatore, 167
- ottimizzazione
- basata sui costi, 167
 - basata sulle regole, 167
 - di una query, (v. query, ottimizzazione)
- outer join, 115
- P**
- partizionamento, 198–199
- orizzontale, 198
 - verticale, 199
- password, 141
- pattern, 108
- permessi, 19, 106, 126–129, 141–142, 159–160, 167
- prima forma normale, 62–64, 66, 78–79
- problemi nella gestione dei dati,
- dati condivisi, 12, 20, 21, 129, 175
 - dati corrotti/persi, 20, 154
 - dati duplicati, 11, 16, 18, 19, 21, 29
 - dati incoerenti, 153, 154, 159, 199–201
 - dati in conflitto, 13, 17–18, 21, 60, 71, 116, 153, 158
 - difficoltà nella variazione dei dati, 13, 14, 17, 18
 - fallimento, 161
- procedure archiviate, 185–188, 196
- prodotto cartesiano, operazione, 37, 39, 42
- programmazione, linguaggi, 178, 180, 194, 202
- proiezione, operazione, 36, 37, 43, 165
- protezione dati, (v. dati, protezione)
- Q**
- query. (v. SQL; SQL, istruzioni)

- query, ottimizzazione, 164–167
- R**
- raggruppamento, 110, 159
 - READ COMMITTED, transazione, 158
 - READ UNCOMMITTED, transazione, 158
 - recupero dati, (v. dati, recupero) registro, 27–28, 34, 48, 148–149
 - relazione, 54, 74
 - E-R (Entità-Relazione), 50–55, 74–77, 81
 - gerarchica, 32, 33, 39, 204
 - molti-a-molti, 55, 74, 75, 81
 - uno-a-molti, 55, 75, 81
 - uno-a-uno, 74, 81
 - REPEATABLE READ, transazione, 158
 - replica, 201–202
 - in sola lettura, 201
 - REVOKE (istruzione), 159, 160, 168
 - riga, 34, 84, 116
 - right outer join, 115
 - ripristino, meccanismi, 20, 147–150, 161–164, 167
 - risorse, 155
 - ROLLBACK (istruzione), 136–137, 150, 153–154, 205
 - roll forward, 149
- S**
- schema, 81
 - concettuale, 81
 - esterno, 81
 - interno, 81
 - scrittura, operazioni, 130, 133–134, 159
 - seconda forma normale, 62, 64, 66–69, 78–79, 82
 - SELECT (istruzione), 93–97, 98, 105, 106, 113, 119, 159
 - selezione, operazione, 37, 39, 43, 47, 48, 165
 - SERIALIZABLE, transazione, 155, 156, 158
 - server, 178–185, 194–197, 205
 - SET TRANSACTION (istruzione), 158, 160
 - sicurezza, 19, 127, 138–142, 159–160, 161–164, 167, 176, 182, 184
 - sort merge (ordina e unisci), metodo, 166
 - sovrascritture non autorizzate, 140
 - SQL (Structured Query Language), 90–92, 106, 116, 124
 - aggregazione, funzione di, 98–100, 110–111
 - caratteri jolly 97, 108
 - condizioni, 95–96, 101, 107–109
 - congiunzione di tabelle, 44, 101–102, 114–115
 - creazione tabelle, 91–92, 103–105, 106, 115–119
 - creazione vista, 117, 160
 - frasi, 94–97, 106, 110–111, 119
 - gestione dati, 90–92, 100, 106, 116
 - GROUP BY (frase), 110
 - HAVING (frase), 111
 - metodi di ricerca, 93–97, 106, 108, 112–115, 163
 - operatori di confronto, 107
 - operatori logici, 107
 - ottimizzazione di una query, 164–167
 - pattern, 108
 - standardizzazione, 124
 - subquery, 112–114
 - Web database, 178–179, 195–196
 - WHERE (frase), 94–97, 106, 110, 119
 - SQL (Structured Query Language), istruzioni, 1–ALL, 159
 - COMMIT, 133, 137, 150, 154, 205
 - CREATE TABLE, 103, 115–119
 - CREATE VIEW, 117
 - DELETE, 104, 116, 118, 119, 159
 - DROP TABLE/DROP VIEW, 118
 - GRANT, 159, 168
 - INSERT, 104, 116, 119, 159
 - LIKE, 97, 108
 - ORDER BY, 98
 - REVOKE, 159, 160, 168
 - ROLLBACK, 136–137, 150, 153–154, 154, 205
 - SELECT, 93–97, 98, 105, 106, 113, 119, 159
 - SET TRANSACTION, 158, 160
 - UPDATE, 104, 116, 119, 159
 - SQL92/SQL99, 124
 - Structured Query Language. (v. SQL)
 - subquery, 112–114
 - subquery correlata, 113–114
 - SUM (funzione), 99, 110
- T**
- tabella/tabelle, 34, 39, 48
 - a due dimensioni, 34, 79
 - base, 160
- U**
- unione (operazione), 37, 39, 40, 48
- u**
- univoco, 30
 - uno-a-molti, relazione, 55, 75, 81
 - uno-a-uno, relazione, 74, 81
- U**
- UPDATE (istruzione), 104, 116, 119, 159
- utente,**
- autorizzazione, 141, 159
 - nome, 141
- V**
- valori funzionalmente dipendenti, 79
 - valori transitivamente dipendenti, 79
- v**
- verticale,
 - distribuzione (v. distribuzione verticale)
 - partizionamento, (v. partizionamento verticale)
- vista, creazione di, 117, 160**
- W**
- Web, database sul, 177–182, 194–197
- WHERE (frase), 94–97, 106, 110, 119**
- X**
- XML (eXtensible Markup Language), 202

L'AUTRICE

Mana Takahashi, dopo una laurea in Economia alla *Tokyo University* si è specializzata nella comunicazione tecnica e ha pubblicato numerosi libri su argomenti quali *Java*, *C*, *XML*, progettazione e amministrazione di sistemi.

UN'AFFASCINANTE GUIDA AI DATABASE. A FUMETTI!



L'IMPERO ORTOFRUTTICOLO DELLA PRINCIPESSA RURUNA E DI CAIN È UN INSIEME DI DATI DUPLICATI E IN CONFLITTO TRA LORO, UN VERO PROBLEMA!! COSA POSSONO FARE? FACILE! DEFINIRE UN DATABASE RELAZIONALE CON L'AUTO DI TICO, LA FATINA DEI DATABASE.

NE **"I MANGA DELLE SCIENZE - DATABASE"** TICO CI INSEGNERÀ A COSTRUIRE UN DATABASE PER LA GESTIONE DELLE VENDITE E DELLE ESPORTAZIONI. IMPAREREMO IL FUNZIONAMENTO DEI DATABASE E IL SIGNIFICATO DI TERMINI COME "SCHEMA", "CHIAVI", "NORMALIZZAZIONE" E "TRANSAZIONI".

INSIEME A RURUNA E CAIN IMPAREREMO A:

- ESTRARRE DATI DA UN DATABASE RELAZIONALE USANDO OPERATORI D'INSIEME E RELAZIONALI;
- APPLICARE IL MODELLO ENTITÀ-RELAZIONE PER RAPPRESENTARE I DATI IN MANIERA ACCURATA;
- GESTIRE I PERMESSI DELL'UTENTE E I BLOCCHI PER IMPEDIRE IL CONFLITTO TRA I DATI O LA LORO DUPLICAZIONE;
- USARE LE ISTRUZIONI SQL PER AGGIORNARE O RECUPERARE DATI E CREARE REPORT.

ARRIVEREMO AD ESPLORARE I FONDAMENTI DI INDICIZZAZIONE, SICUREZZA, RECUPERO, REPLICAZIONE DEI DATABASE E ALTRO ANCORA.

SE QUANDO SENTITE PARLARE DI "DATABASE" VI SEMBRA DI PERDERVI IN UN LABIRINTO DI NUMERI E DATI FUORI DAL VOSTRO CONTROLLO, FATE COMPAGNIA A RURUNA E CAIN MENTRE IMPARANO TUTTO CIÒ DI CUI HANNO BISOGNO NE **"I MANGA DELLE SCIENZE - DATABASE"**.



la Repubblica Le Scienze



Pubblicazione settimanale da vendersi esclusivamente
in abbinamento a la Repubblica oppure a Le Scienze.
Supplemento al numero in edicola.

9,90 euro + il prezzo di Repubblica oppure di Le Scienze.