FONDAMENTI DI BASI DI DATI

Soluzione degli esercizi

Antonio Albano, Giorgio Ghelli, Renzo Orsini

5 marzo 2019

INDICE

Capitolo 1

SISTEMI PER BASI DI DATI

Esercizio 1.1 Discutere le differenze tra i modi di descrivere i dati in un sistema per la gestione di basi di dati e in un sistema di archiviazione.

Soluzione 1.1 Vedi testo.

Esercizio 1.2 Elencare alcune domande (fra cinque e dieci) da fare ad un produttore di sistemi per la gestione di dati al fine di stabilire se il sistema che propone può essere classificato come un sistema per la gestione basi di dati centralizzate.

Soluzione 1.2
Un insieme di domande per un sistema relazionale potrebbe essere:

- Posso usare SQL da un programma per accedere ai dati, definire nuovi insiemi di dati o nuovi vincoli?
- Quali sono i vincoli d'integrità che il sistema controlla? Ad esempio posso cancellare uno studente se ci sono dei suoi esami sostenuti?
- Se si rompe il disco che contiene la base di dati, questa può essere recuperata?
- Quanti utenti che eseguono la stessa operazione nello stesso momento può reggere il sistema?
- Posso aggiungere dischi mentre il sistema è in funzione? e posso modificare la struttura di una tabella?
- Posso proteggere solo alcune righe di una tabella dall'accesso di certi utenti?
- Posso interrogare il sistema con SQL per sapere quante sono le tabelle dell'utente Caio?

Esercizio 1.3 Discutere vantaggi e svantaggi di un sistema per la gestione di basi di dati.

Soluzione 1.3 Vedi testo.

Esercizio 1.4 Spiegare la differenza tra i seguenti termini: base di dati e sistema per la gestione di basi di dati.

Soluzione 1.4 Da Fare.

Esercizio 1.5 Discutere i concetti di indipendenza logica e fisica, confrontandoli con i concetti di modulo e tipo di dato astratto dei linguaggi di programmazione.

Soluzione 1.5 Da Fare.

Esercizio 1.6 Discutere le differenze tra il modello dei dati di un sistema di archiviazione e un sistema per basi di dati.

Soluzione 1.6 Da Fare.

Esercizio 1.7 Discutere i compiti del programmatore delle applicazioni e dell'amministratore della base di dati.

Soluzione 1.7 Da Fare.

Esercizio 1.8 Quali delle seguenti affermazioni è vera?

a) Sistema informatico è un sinonimo di sistema informativo.
b) Un linguaggio di interrogazione richiede la conoscenza di un linguaggio di programmazione.
c) Per usare correttamente una base di dati l'utente (persona o programma) deve conoscere l'organizzazione fisica dei dati.
d) L'organizzazione fisica di una base di dati va programmata dall'amministratore della base di dati.
e) Schema logico, schema fisico e schema esterno sono sinonimi.
f) Per soddisfare le esigenze degli utenti delle applicazioni non occorre un linguaggio di programmazione.
g) Le transazioni nei sistemi per basi di dati hanno le stesse proprietà dei programmi nei linguaggi con archivi.
h) Per realizzare un sistema informatico il personale tecnico realizza innanzitutto applicazioni che usano basi di dati.
i) Il programmatore delle applicazioni decide quali sono i dati accessibili agli utenti.

Soluzione 1.8
Vere: d), h). **False**: a), b), c), e), f), g), i).

Capitolo 2

I MODELLI DEI DATI

Esercizio 2.1 Discutere le differenze tra i modi in cui si rappresentano le associazioni fra insiemi di dati nei modelli dei dati ad oggetti, entità-relazioni e relazionale.

Soluzione 2.1 Vedi testo.

Esercizio 2.2 Discutere le differenze tra le nozioni di tipo oggetto e di classe.

Soluzione 2.2 Vedi testo.

Esercizio 2.3 Discutere le differenze tra le nozioni di tipo oggetto definito per ereditarietà e di sottoclasse.

Soluzione 2.3 Vedi testo.

Esercizio 2.4 Discutere i vincoli che si possono descrivere graficamente con il modello ad oggetti.

Soluzione 2.4 Vedi testo.

Esercizio 2.5 Si definisca uno schema grafico per rappresentare con il modello a oggetti tre insiemi di entità: le persone e i sottoinsiemi delle persone viventi e delle persone decedute. Dare delle proprietà interessanti per gli elementi di questi insiemi. Dire quali problemi si presenterebbero nel modellare questi insiemi se il modello non offrisse né il meccanismo dei tipi oggetti definiti per ereditarietà, né il meccanismo delle sottoclassi.

Soluzione 2.5 Da fare.

Esercizio 2.6 Si vuole automatizzare il sistema di gestione degli animali in uno zoo. Ogni esemplare di animale ospitato è identificato dal suo genere (ad esempio, zebra) e da un codice unico all'interno del genere di appartenenza. Per ogni esemplare si

memorizzano la data di arrivo nello zoo, il nome proprio, il sesso, il paese di provenienza e la data di nascita. Lo zoo è diviso in aree; in ogni area c'è un insieme di case, ognuna destinata ad un determinato genere di animali. Ogni casa contiene un insieme di gabbie, ognuna contenente un solo esemplare. Ogni casa ha un addetto che pulisce ciascuna gabbia in un determinato giorno della settimana. Gli animali sono sottoposti periodicamente a controllo veterinario; in un controllo, un veterinario rileva il peso degli esemplari, diagnostica un'eventuale malattia e prescrive il tipo di dieta da seguire.

Dare uno schema grafico della base di dati.

Soluzione 2.6 Da fare.

Esercizio 2.7 Una banca gestisce informazioni sui mutui dei propri clienti e le rate del piano di ammortamento. Un cliente può avere più di un mutuo. Ai clienti viene inviato periodicamente un resoconto sulle rate pagate del tipo mostrato in figura. Per le rate pagate dopo la data di scadenza è prevista una mora. Si progetti la base di dati e successivamente si modifichi lo schema per trattare anche il fatto che i clienti fanno prima una richiesta di mutuo che poi può essere concesso con un rimborso a rate secondo un certo piano di ammortamento.

RESOCONTO MUTUO

Codice mutuo: 250 **Data**: 07/01/2005
Scadenza: 01/01/2008
Ammontare: 70 000,00
Codice cliente: 2000
Nome cliente: Mario Rossi
Indirizzo: Via Roma, 1 Pisa

Numero rata	Scadenza	Ammontare	Data Versamento
1	1/07/05	6 000,00	29/06/05
2	1/01/06	6 000,00	30/12/05
3	1/07/06	6 100,00	29/06/06
4	01/01/07	6 100,00	30/12/06

Soluzione 2.7 Il progetto concettuale iniziale è in Figura 2.1, mentro quello finale è in Figura 2.2.

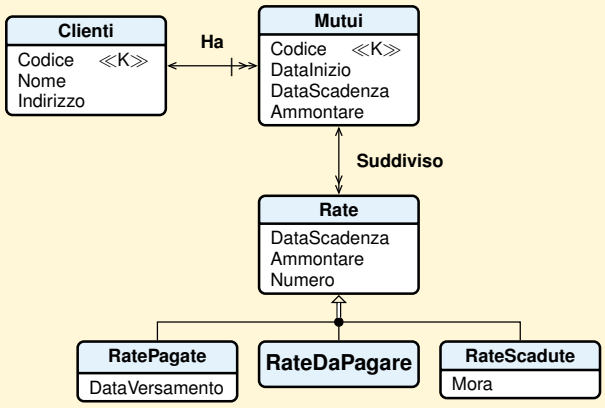


Figura 2.1: Gestione dei mutui: progetto concettuale iniziale

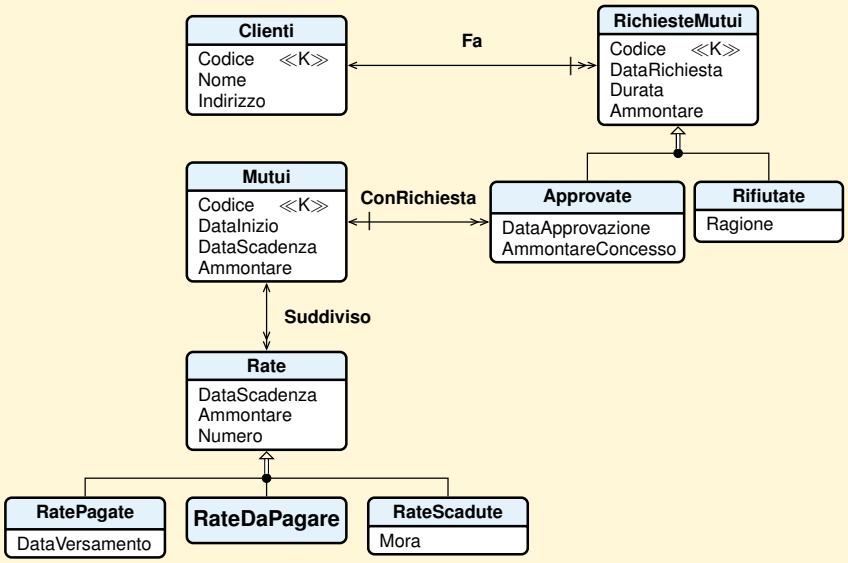


Figura 2.2: Gestione dei mutui: progetto concettuale finale

Esercizio 2.8 Si vogliono trattare informazioni su attori e registi di film. Di un attore o un regista interessano il nome, che lo identifica, l'anno di nascita e la nazionalità. Un attore può essere anche un regista. Di un film interessano il titolo, l'anno di produzione, gli attori, il regista e il produttore. Due film prodotti lo stesso anno hanno titolo diverso.

Dare uno schema grafico della base di dati.

Soluzione 2.8 Il progetto concettuale è in Figura 2.3.

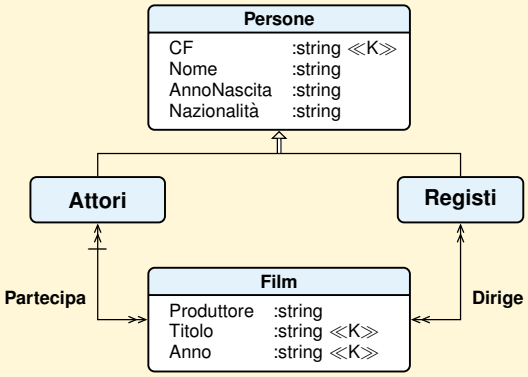


Figura 2.3: Gestione di dati su film: progetto concettuale

Esercizio 2.9 Un'azienda vuole gestire le informazioni riguardanti gli impiegati, i dipartimenti e i progetti in corso.
Di un impiegato interessano il codice, assegnato dall'azienda, che l'identifica, il nome e cognome, l'anno di nascita, il sesso e i familiari a carico, dei quali interessano il nome, il sesso, la relazione di parentela e l'anno di nascita.
Di un dipartimento interessano il numero, che lo identifica, il nome, la città dove si trova.
Di un progetto interessano il numero, che lo identifica, e il codice. Un progetto è gestito da un solo dipartimento.
Gli impiegati afferiscono ad un dipartimento, che gestisce più progetti ed è diretto da un impiegato. Gli impiegati partecipano a più progetti, che si svolgono presso dipartimenti di città diverse, ad ognuno dei quali dedicano una percentuale del proprio tempo. Gli impiegati sono coordinati da un responsabile, che è un impiegato. Dei direttori e dei responsabili interessa l'anno di nomina.
Dare uno schema grafico della base di dati.

Soluzione 2.9 Il progetto concettuale è in Figura 2.4.

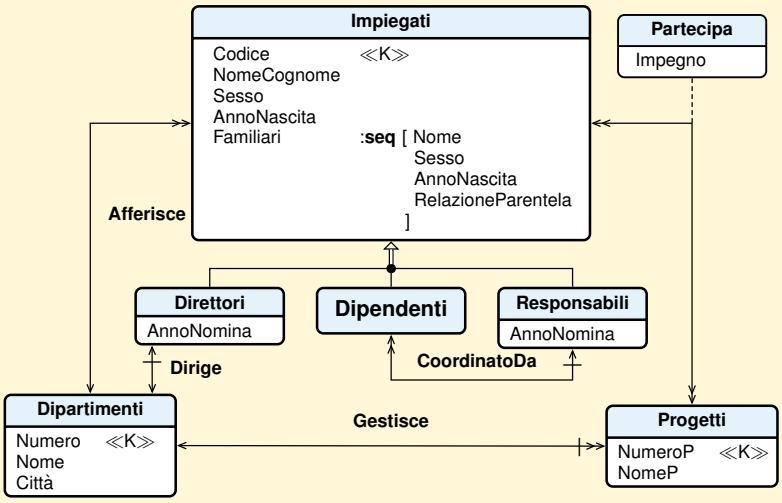


Figura 2.4: Gestione di dati aziendali: progetto concettuale

Capitolo 3

LA PROGETTAZIONE DI BASI DI DATI

Esercizio 3.1 Condomìni
Si supponga di dover memorizzare in una base di dati le informazioni di interesse per un amministratore di condomìni. Di un condominio interessano l'indirizzo e il numero del conto corrente dove vengono fatti i versamenti per le spese sostenute. Un condominio si compone di un certo numero di appartamenti, dei quali interessano il numero dell'interno, il numero dei vani, la superficie, lo stato (libero od occupato).

Gli appartamenti possono essere locati, e dell'inquilino interessano il nome, il codice fiscale, il telefono e il saldo, cioè la somma che l'inquilino deve all'amministrazione condominiale per le spese sostenute. Alcuni appartamenti locati possono essere stati disdetti, ed in questo caso interessa la data della disdetta.

Un appartamento può avere più proprietari, e un proprietario può possedere più appartamenti. Di ogni proprietario interessano il nome, il codice fiscale, l'indirizzo, il telefono e il saldo, cioè la somma che il proprietario deve all'amministrazione condominiale per le spese sostenute.

Le spese riguardano i condomìni e di esse interessano il codice di identificazione, la natura (luce, pulizia, ascensore ecc.), la data e l'importo. Fra le spese si distinguono quelle straordinarie, a carico dei proprietari, e quelle ordinarie, a carico degli inquilini. Le spese ordinarie vengono pagate in un'unica rata, mentre le spese straordinarie possono essere pagate in più rate e di ognuna di esse occorre ricordare la data e l'importo.

Progettare lo schema della base di dati e dare la specifica delle operazioni per l'immissione dei dati.

Soluzione 3.1
Il progetto concettuale è in Figura 3.1.

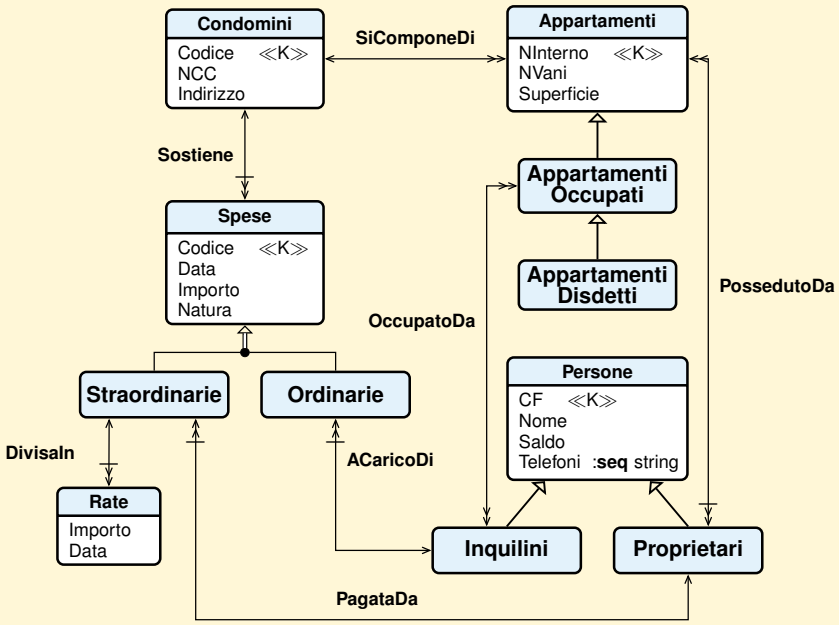


Figura 3.1: Gestione dei dati di condomini: progetto concettuale

Esercizio 3.2 Società Mega S.p.A.
Si vogliono gestire informazioni riguardanti gli impiegati, le loro competenze, i progetti a cui partecipano e i dipartimenti a cui appartengono.
Ogni impiegato ha una matricola che lo identifica, assegnata dalla società. Di ogni impiegato interessano il nome, la data di nascita e la data di assunzione. Se un impiegato è coniugato con un altro dipendente della stessa società, interessano la data del matrimonio e il coniuge. Ogni impiegato ha una qualifica (ad esempio, segretaria, impiegato, programmatore, analista, progettista ecc.). Dei laureati e delle segretarie interessano altre informazioni. Dei laureati interessa il tipo di laurea e delle segretarie le lingue straniere conosciute.
La società è organizzata in dipartimenti, identificati da un nome e da un numero di telefono. Un impiegato afferisce ad un solo dipartimento. Ogni dipartimento si approvvigiona presso vari fornitori e un fornitore può rifornire più dipartimenti. Di ogni fornitore interessano il nome e l'indirizzo. Interessano, inoltre, la data e il fornitore dell'ultimo acquisto fatto da un dipartimento.
La società lavora a diversi progetti, ciascuno dei quali è localizzato in una città. Più impiegati partecipano ad un progetto e un impiegato può partecipare a più progetti, ma tutti localizzati sulla stessa città. Di ogni città con un progetto in corso interessano la sua popolazione e la regione. Un impiegato può avere più competenze, ma usarne solo alcune per un particolare progetto. Un impiegato usa ogni sua competenza in almeno un progetto. Ad ogni competenza è assegnato un codice unico e una descri-

zione. I progetti in corso sono identificati da un numero ed interessa una stima del loro costo.
Progettare lo schema della base di dati e dare la specifica delle operazioni per l'immissione dei dati.

Soluzione 3.2 Da Fare.

Esercizio 3.3 Anagrafe
Si vogliono trattare informazioni sulle persone che vivono o sono decedute in un comune italiano.
Di una persona interessano: nome, cognome, codice fiscale, data di nascita (giorno, mese, anno), età, indirizzo (via, numero, cap, comune), sesso (m, f), stato civile (celibe (nubile), coniugato(a), vedovo(a), separato(a), divorziato(a), deceduto(a)), madre, padre e antenati.
Una persona può essere vivente o deceduta. Di una persona vivente interessano: indirizzo, numeri telefonici, comune di residenza, familiari conviventi, figli viventi e figli conviventi. Le persone di un nucleo familiare condividono lo stesso indirizzo, telefono e comune di residenza.
Di una persona deceduta interessano: data del decesso, età, comune del decesso, comune dove è stata seppellita.
Di un matrimonio interessano: data, sposo, sposa e comune dove è stato celebrato. Non sono ammessi matrimoni fra consanguinei ovvero fra persone che hanno uno stesso antenato.
Di un comune interessano: nome, se capoluogo di provincia, prefisso telefonico, gli abitanti, il numero degli abitanti, le persone seppellite e decedute, il numero delle persone seppellite e decedute.
Vanno previste le seguenti operazioni per le quali vanno fissati i parametri e il tipo del risultato:

a) creazione di una persona nubile o celibe;
b) nascita di un figlio;
c) matrimonio;
d) antenati viventi di una persona;
e) nome e cognome dei genitori di una persona;
f) nome, cognome e relazione di parentela dei familiari conviventi di una persona;
g) cambio di residenza di una persona e dei suoi familiari conviventi;
h) una persona e i suoi conviventi vanno a vivere con un'altra persona;
i) una persona va a vivere da sola;
j) decesso di una persona.

Soluzione 3.3 Da Fare.

Esercizio 3.4 Ufficio della motorizzazione
Si vogliono modellare i seguenti fatti di interesse di un ipotetico Ufficio della motorizzazione.

Produttori di automobili C'è un certo numero di produttori di automobili, ciascuno identificato da un nome (FIAT, FORD ecc.). I dati di nuovi produttori possono essere immessi in ogni momento, se il produttore ha l'autorizzazione ad iniziare l'attività commerciale.
L'autorizzazione non può essere ritirata e non più di cinque produttori possono essere in attività contemporaneamente. Un produttore è considerato attivo finché possiede automobili registrate come prodotte da lui e non ancora vendute; nel momento in cui un produttore non possiede auto, il suo permesso di operare può essere sospeso. I dati di un produttore vengono eliminati solo quando viene eliminata la storia di tutte le auto da lui prodotte.

Automobili Un'automobile è caratterizzata da un modello, dall'anno di produzione, da un numero di serie assegnatogli dal produttore, unico fra le automobili da lui prodotte. I dati di un'automobile vengono immessi all'atto della sua registrazione presso l'Ufficio della Motorizzazione. Al momento della registrazione, all'automobile viene assegnato un numero, unico per ciascuna automobile e non modificabile, e la data di registrazione. Il produttore viene registrato come primo proprietario.
Un'automobile può essere registrata in qualsiasi giorno dell'anno in cui è stata costruita, ma può essere registrata solo entro il 31 gennaio se costruita l'anno precedente. Nel caso di distruzione, viene registrata la data di distruzione, e da questo momento l'automobile non può essere più trasferita. Infine, la storia di un'automobile va conservata per due anni dopo la sua distruzione.

Modelli di automobile Ogni automobile è caratterizzata da un modello (Panda, Uno, Escort ecc.). Le automobili di ciascun modello sono prodotte dallo stesso produttore, il quale è libero di introdurre nuovi modelli sul mercato in qualsiasi momento. Il nome di ciascun modello è unico fra tutti i modelli registrati. Le automobili di uno stesso modello hanno lo stesso consumo di benzina. Un modello ha una potenza di almeno 6 cavalli e una cilindrata compresa fra 400 e 3.000 cc.
I dati su un modello vanno conservati finché esiste nella base di dati un'automobile di tale modello. Le automobili di un certo modello non possono essere registrate se tale modello non è ancora noto all'Ufficio della motorizzazione.

Rivenditori I rivenditori sono preposti alla distribuzione di automobili nuove, o usate, ai privati. Di un rivenditore interessano il nome, l'indirizzo, il telefono e l'eventuale numero del fax. Nuovi rivenditori possono sorgere in ogni momento, ma la loro attività commerciale può iniziare solo se hanno ricevuto il permesso dagli uffici competenti. Un rivenditore può trattare automobili nuove di al più tre produttori diversi.

Ogni rivenditore è considerato operante finché possiede automobili; in caso contrario può richiedere la sospensione del permesso di operare. I dati di un rivenditore non operante vengono eliminati solo se questo non è stato proprietario di un'auto di cui si conserva la storia.

Privati I privati sono persone proprietarie di una o più automobili già registrate. Di un privato interessano il nome, l'indirizzo e il telefono. I dati dei privati vengono immessi con l'acquisto della prima automobile, ed eliminati solo se essi non sono stati proprietari di un'automobile di cui si conserva la storia.

Trasferimenti di proprietà In ogni momento un'automobile può essere posseduta: dal suo produttore (automobile invenduta), da un rivenditore, oppure da un gruppo di privati.
All'atto del trasferimento della proprietà di un'automobile vengono registrate le seguenti informazioni: un codice che identifica il trasferimento, la data di trasferimento, l'automobile trasferita, il vecchio e il nuovo proprietario.
Vi sono norme che vincolano il trasferimento di un'automobile:

- un'automobile distrutta non può essere trasferita;
- un'automobile può essere venduta da un produttore solo ad un rivenditore, e un produttore non può acquistare automobili;
- un'automobile può essere venduta da un rivenditore solo a privati o gruppi di privati.

I dati su un trasferimento possono essere eliminati solo quando l'automobile cessa di essere di interesse per l'Ufficio della Motorizzazione.
Vanno previste le seguenti operazioni.

- Registrazione di una nuova auto.
- Distruzione di un'auto.
- Registrazione della vendita di un'auto.
- Eliminazione della storia delle auto distrutte da almeno due anni.
- Registrazione di un nuovo produttore in attesa del permesso di operare.
- Autorizzazione di un produttore ad operare.
- Sospensione delle attività di un produttore.
- Eliminazione di un produttore non operante.
- Registrazione di un nuovo modello.
- Registrazione di un nuovo rivenditore.
- Sospensione delle attività di un rivenditore.
- Eliminazione di un rivenditore non operante.

Soluzione 3.4 Da Fare.

Esercizio 3.5 Organizzazione di una conferenza
Si vogliono trattare i dati riguardanti le *Working Conferences* dell'IFIP (*International Federation for Information Processing*). Una IFIP *Working Conference* è una conferenza internazionale intesa a riunire esperti di tutti i paesi aderenti all'IFIP per discutere problemi che interessano uno o più IFIP *Working Group.* Ogni *Working Group* opera sotto gli auspici di un *Technical Committee* costituito dai rappresentanti nazionali dei paesi aderenti all'IFIP.
Alla conferenza possono partecipare solo persone che hanno ricevuto un invito. L'invito è inviato a tutti i membri dei *Working Groups* e *Technical Committees* interessati. Il numero delle persone che parteciperanno ai lavori deve essere superiore ad una soglia minima, per garantirsi la copertura dei costi, ed inferiore ad una soglia massima, per non superare le capacità ricettive delle strutture.
La conferenza è organizzata da due comitati: il Comitato di Programma e il Comitato Organizzatore. Il primo cura gli aspetti scientifici della conferenza: nomina il Comitato dei Revisori, che esaminerà gli articoli sottoposti alla conferenza, e decide quali articoli accettare, non superando il numero massimo prestabilito.
Il Comitato Organizzatore cura gli aspetti finanziari, logistici, gli inviti e la pubblicità.
Ogni comitato è costituito da esperti ed è previsto un *Chairman* per ogni comitato ed un *General Chairman* per la conferenza. Tutti i comitati lavorano utilizzando dati comuni che vanno raccolti ed elaborati in modo consistente.

Procedure da automatizzare L'applicazione da realizzare ha lo scopo di agevolare le attività di entrambi i comitati, pensati come due settori aziendali che operano utilizzando dati in comune. I comitati devono svolgere le seguenti attività.

Comitato di Programma

- Preparare la lista degli esperti a cui sollecitare la presentazione di un articolo.
- Registrare le lettere d'intenti, cioè le risposte date da coloro che intendono inviare un lavoro. Ogni esperto invia al più una lettera che verrà presa in considerazione solo se perviene entro la data prestabilita. I lavori con nessun autore fra gli esperti che hanno risposto con una lettera d'intenti avranno una priorità bassa, se il numero complessivo dei lavori da esaminare supera il massimo prestabilito.

Comitato Organizzatore

- Preparare la lista degli esperti da invitare alla conferenza. Nella lista devono essere inclusi: i membri di tutti i *Technical Committees* e *Working Groups* interessati; i membri del Comitato dei Revisori e tutti coloro che hanno proposto un articolo. Occorre evitare di mandare alla stessa persona più di un invito.
- Registrare le adesioni alla conferenza pervenute entro la data prefissata.
- Generare la lista finale dei partecipanti alla conferenza. Se le adesioni superano il numero massimo prestabilito, verrà data priorità, nell'ordine, ai membri dei *Technical Committees*, ai membri dei *Working Groups* e del Comitato dei Revisori, agli autori degli articoli ammessi alla conferenza, agli autori degli articoli rifiutati.
- Registrare gli articoli proposti per la conferenza e pervenuti entro una data prefissata.
- Distribuire i lavori fra i membri del Comitato dei Revisori. Ogni lavoro sarà revisionato da 3 a 5 revisori, a seconda del numero complessivo dei lavori pervenuti.
- Raccogliere i pareri dei revisori e selezionare il numero prefissato di lavori da presentare alla conferenza.
- Raggruppare i lavori selezionati in sessioni e scegliere un *Chairman* per ogni sessione fra i membri del Comitato di Programma.

Soluzione 3.5 Da Fare.

Capitolo 4

IL MODELLO RELAZIONALE

Esercizio 4.1 Elencare le differenze tra la nozione di ennupla nei sistemi relazionali e la nozione di oggetto nei sistemi ad oggetti.

Soluzione 4.1
Le ennuple del modello relazionale hanno campi di tipo elementare, non hanno una nozione di identità, non hanno componenti procedurali come i metodi degli oggetti, non hanno nozioni di inclusione, eredit, incapsulazione. Dal punto di vista della modellazione, ne consegue, in particolare, che:

- nel modello relazionale non è possibile rappresentare le entità della realtà con un'unica ennupla se queste hanno una struttura complessa (ad esempio, se hanno attributi multivalore);
- nel modello relazionale le associazioni si rappresentano usando il meccanismo delle chiavi esterne;
- nel modello relazionale è necessario aggiungere un campo chiave per distinguere entità con gli stessi valori per tutti gli attributi.

Esercizio 4.2 Si traduca lo schema concettuale ottenuto dall'Esercizio 2.7 in uno schema relazionale.

Soluzione 4.2
In Figura 4.1 si ripete la soluzione dell'Esercizio 2.7. Lo schema relazionale è mostrato in Figura 4.2.

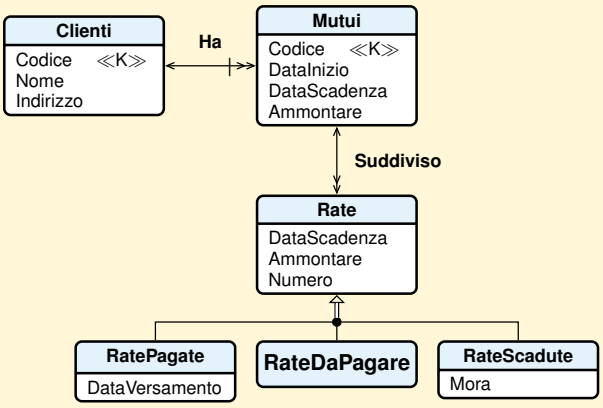


Figura 4.1: Schema concettuale Mutui, Esercizio 2.7

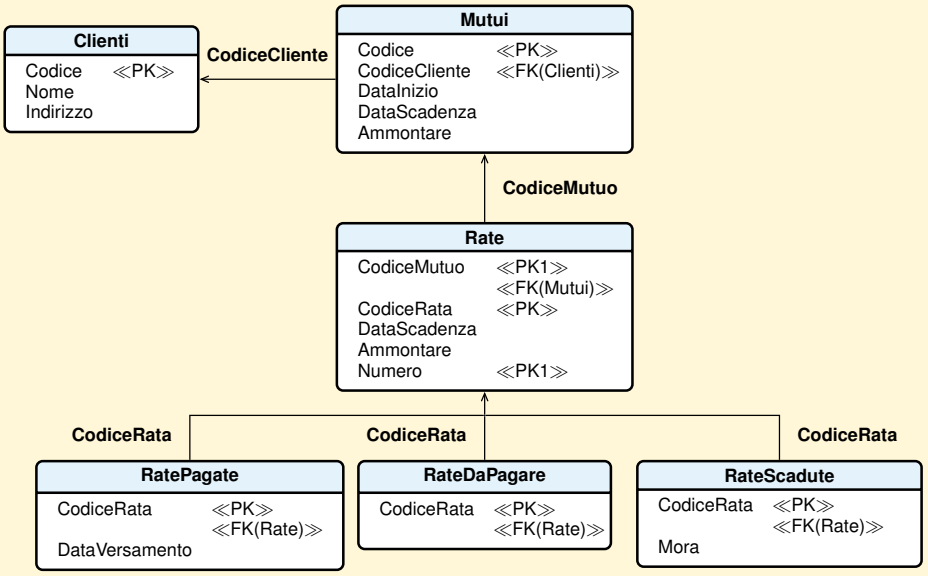


Figura 4.2: Gestione dei mutui: progetto relazionale

Esercizio 4.3 Si traduca lo schema concettuale ottenuto dall'Esercizio 2.9 in uno schema relazionale.

Soluzione 4.3
In Figura 4.3 si ripete la soluzione dell'Esercizio 2.9. Lo schema relazionale è mostrato in Figura 4.4.

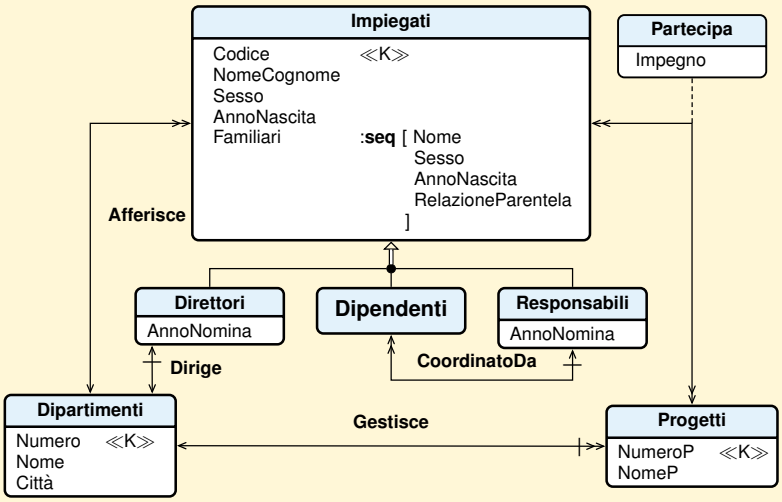


Figura 4.3: Schema concettuale Azienda, Esercizio 2.9

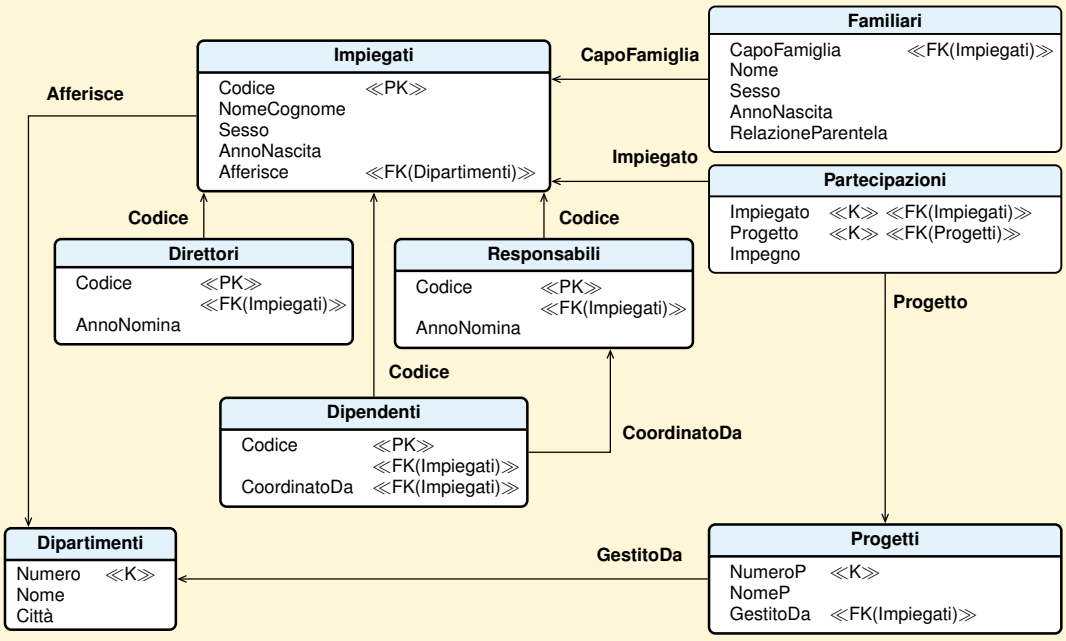


Figura 4.4: Gestione di dati aziendali: progetto relazionale

Esercizio 4.4 Dimostrare le seguenti proprietà:

a) $\sigma_{\phi \wedge \psi}(E) = \sigma_{\phi}(E) \cap \sigma_{\psi}(E)$;
b) $\sigma_{\phi_1}(\sigma_{\phi_2}(E)) = \sigma_{\phi_1 \wedge \phi_2}(E)$;
c) $\sigma_{\phi}(E_1 \cup E_2) = \sigma_{\phi}(E_1) \cup \sigma_{\phi}(E_2)$;
d) $\pi_Y(\pi_X(E)) = \pi_Y(E)$, se $Y \subseteq X$;
e) $\pi_X(\sigma_{\phi}(E) = \sigma_{\phi}(\pi_X(E))$, se ϕ usa solo attributi in X;
f) $\sigma_{\phi}(E_1 \underset{\phi_J}{\bowtie} E_2) = \sigma_{\phi}(E_1) \underset{\phi_J}{\bowtie} E_2$, se ϕ usa solo attributi di E_1.
g) $\sigma_{\phi}({}_A\gamma_F(E)) =_A \gamma_F(\sigma_{\phi}(E))$, se ϕ usa solo attributi in A.

Soluzione Da fare.

Esercizio 4.5 Si consideri lo schema relazionale

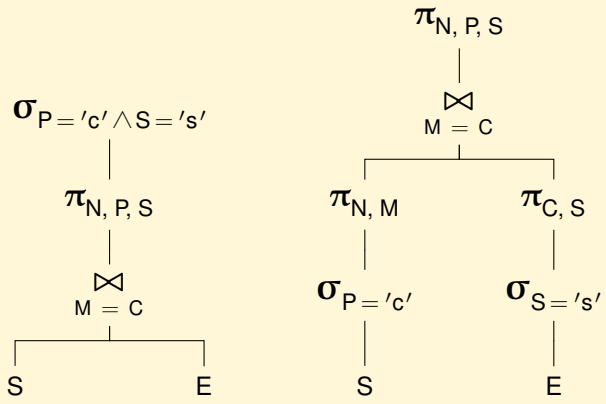
S(M, N, P, A)
E(C, S, V)

e l'espressione dell'algebra relazionale

$\sigma_{P='c' \wedge S='s'}(\pi_{N,P,S}(S \underset{M=C}{\bowtie} E))$.

Si dia una rappresentazione ad albero dell'interrogazione e si mostri come si modifica l'albero dopo la riscrittura algebrica.

Soluzione 4.4



(a) *Albero iniziale* **(b)** *Albero riscritto*

Esercizio 4.6 Si consideri lo schema relazionale dell'esercizio precedente e l'espressione dell'algebra relazionale

$$\pi_{N,S}(\sigma_{A='a' \wedge S='s'}(\pi_{N,M,A}(S) \underset{M=C}{\bowtie} \pi_{C,S}(E))).$$

Si dia una rappresentazione ad albero dell'interrogazione e si mostri come si modifica l'albero dopo la riscrittura algebrica.

Soluzione 4.5 Da Fare.

Capitolo 5

NORMALIZZAZIONE DI SCHEMI RELAZIONALI

Esercizio 5.1 Usando gli assiomi di Armstrong si dimostri che:

a) se $X \to YZ$, allora $X \to Y$ e $X \to Z$;
b) se $X \to Y$ e $WY \to Z$, allora $XW \to Z$;
c) se $X \to YZ$ e $Z \to BW$, allora $X \to YZB$.

Soluzione 5.1

a) Dimostriamo che $F \vdash X \to YZ$ implica $F \vdash X \to Y$. $F \vdash YZ \to Y$ per riflessività, e la tesi segue per transitività. $F \vdash X \to YZ \Rightarrow F \vdash X \to Z$ è analogo.
b) Dimostriamo che se $F \vdash X \to Y$ (1) e $F \vdash WY \to Z$ (2), allora $F \vdash XW \to Z$. Per arricchimento, da (1) segue $F \vdash XW \to YW$ (3); la tesi segue per transitività da (3) e (2).
c) Dimostriamo che se $F \vdash X \to YZ$ (1) e $F \vdash Z \to BW$ (2), allora $F \vdash X \to YZB$. Per arricchimento, da (2) segue $F \vdash YZ \to YBW$ (3); la tesi segue per transitività da (1) e (3).

Esercizio 5.2 Sia $F = \{A \to B, C \to B, D \to ABC, AC \to D\}$ un insieme di dipendenze funzionali. Trovare un insieme di dipendenze G in forma canonica equivalente a F.

Soluzione 5.2
Sia $F = \{A \to B, C \to B, D \to ABC, AC \to D\}$. Il calcolo di C^+ ed A^+ mostra che non sono presenti attributi estranei. La dipendenza $D \to B$ è la sola dipendenza ridondante. Lo schema in forma canonica è quindi: $G = \{A \to B, C \to B, D \to A, D \to C, AC \to D\}$.

Esercizio 5.3 Sia $R\langle T, F\rangle$ uno schema relazionale. Dimostrare che se un attributo $A \in T$ non appare a destra di una dipendenza in F, allora A appartiene ad ogni chiave di R.

Soluzione 5.3
Consideriamo un qualunque Y tale che $A \notin Y$. Esaminando il funzionamento dell'algoritmo di chisura lenta, si osserva che A non può appartenere ad Y^+, per cui $A \notin Y$ implica che Y non può essere superchiave, per cui Y non può neppure essere una chiave.

Esercizio 5.4 Sia $R(A, B, C, D, E)$ uno schema di relazione su cui siano definite le dipendenze funzionali:

$$F = \{AB \to CDE, AC \to BDE, B \to C, C \to B, C \to D, B \to E\}$$

Si richiede di:

a) portare F in forma canonica;
b) determinare le possibili chiavi;
c) mostrare che lo schema non è in terza forma normale;
d) portare lo schema in terza forma normale.

Soluzione 5.4

$$F = \{AB \to CDE, AC \to BDE, B \to C, C \to B, C \to D, B \to E\}$$

a) Una copertura canonica per F è data da:

$$G = \{B \to C, B \to D, B \to E, C \to B\}.$$

b) Ogni chiave deve contenere A, poiché A non è implicata da altri attributi. Quindi, dato che $AB^+ = T$, AB è chiave. Dato che $AC^+ = T$, AC è chiave. Calcolando AD^+, AE^+ e ADE^+, si verifica che non ci sono altre chiavi.
c) Lo schema non è in 3NF a causa delle dipendenze $C \to D$, $B \to E$, poiché D ed E non sono attributi primi e C e B non sono chiavi.
d) Applicando l'algoritmo di sintesi a G otteniamo la seguente decomposizione, che non è comunque la sola possibile: $\{BCDE, AB(\text{o } AC)\}$. Si noti che AB (o AC) va aggiunto perché altrimenti lo schema risultante non conterrebbe alcuna relazione che contenga una chiave dello schema originale (e inoltre andrebbe perduto l'attributo A).

Esercizio 5.5 Mostrare (*a*) che se uno schema relazionale è in BCNF allora è anche in 3NF e (*b*) che se uno schema non è in 3NF allora non è neanche in BCNF.

Soluzione 5.5
(*a*): Per definizione, se uno schema relazionale è in BCNF allora per ogni $X \to A \in F^+$ non banale X è una superchiave, per cui lo schema è in 3NF.
(*b*) equivale ad (*a*) per contrapposizione.

Esercizio 5.6 Si consideri l'insieme di attributi $T = ABCDEGH$ e l'insieme di dipendenze funzionali $F = \{AB \to C, AC \to B, AD \to E, B \to D, BC \to A, E \to G\}$. Per ognuno dei seguenti insiemi di attributi X, (*a*) trovare una copertura della proiezione di F su X, e (*b*) dire qual è la forma normale più forte soddisfatta da X:

a) $X = ABC$;
b) $X = ABCD$;
c) $X = ABCEG$;
d) $X = ABCH$;
e) $X = ABCDEGH$.

Soluzione 5.6
Dobbiamo proiettare $F = \{AB \to C, AC \to B, AD \to E, B \to D, BC \to A, E \to G\}$ su

a) $X = ABC$;
b) $X = ABCD$;
c) $X = ABCEG$;
d) $X = ABCH$;
e) $X = ABCDEGH$.

Sia K l'insieme degli attributi che non appaiono a destra di nessuna dipendenza e sia U l'insieme degli attributi che non appaiono a sinistra di nessuna dipendenza; in questo caso $K = H$ e $U = GH$. Per proiettare le dipendenze su X calcoliamo Y^+ per ogni sottoinsieme stretto non vuoto di $X - U$, e, se $Y^+ \neq Y$, aggiungiamo $Y \to (X \cap Y^+ - Y)$ alle dipendenze della proiezione. Consideriamo per prima cosa i sottoinsiemi più piccoli e, ogni volta che scopriamo che $A \in Y^+$, ignoriamo tutti i soprainsiemi di YA, dato che A sarebbe estraneo nella dipendenza; indichiamo (*) sotto per indicare gli insiemi ignorati per questo motivo. In particolare, se troviamo un Y' tale che $Y'^+ = (X - K)$, allora non consideriamo più nessun soprainsieme di Y'.
Tra i singoletti, gli unici con chiusura non banale sono $B^+ = BD$ ed $E^+ = EG$.
Passando alle coppie abbiamo: $AB^+ = ABCDEG$, $AC^+ = ACBDEG$, $AD^+ = ADEG$, $AE^+ = AEG(**)$, $BC^+ = BCDAEG$, $BD^+ = (*)$, $BE^+ = BEDG(**)$, $CD^+ = CD(**)$, $CE^+ = CEG(**)$, $DE^+ = DEG(**)$. Le chiusure segnate con (**) non sono interessanti perché banali o perché il determinante può essere diviso in due sottoinsiemi Y' ed Y'' tali che $(Y' \cup Y'')^+ = Y'^+ \cup Y''^+$. Dato che AB^+,

AC^+ e BC^+ contengono $X - K$, è inutile considerarne i soprainsiemi. Quindi, le sole terne da considerare sono ADE, BDE e CDE, utili nel caso e). $ADE^+ = (*)$, $BDE^+ = (*)$, $CDE^+ = CDEG(**)$. Possiamo ora calcolare una copertura per le proiezioni, considerando, per ciascun $Y \subset X$ con $(X \cap Y^+ - Y) \neq \emptyset$, la dipendenza $Y \to (X \cap Y^+ - Y)$. Consideriamo solo gli Y la cui chiusura sia "interessante" come sopra specificato, ovvero consideriamo solo: $B^+ = BD$, $E^+ = EG$, $AB^+ = ABCDEG$, $AC^+ = ACBDEG$, $AD^+ = ADEG$, $BC^+ = BCDAEG$.

a) $X = ABC$: $\{AB \to C, AC \to B, BC \to A\}$;
b) $X = ABCD$: $\{B \to D, AB \to CD, AC \to BD, BC \to AD\}$;
c) $X = ABCEG$: $\{E \to G, AB \to CEG, AC \to BEG, BC \to AEG\}$;
d) $X = ABCH$: $\{AB \to C, AC \to B, BC \to A\}$;
e) $X = ABCDEGH$: $\{B \to D, E \to G, AB \to CDEG, AC \to BDEG, BC \to ADEG, AD \to EG\}$.

Le coperture così ottenute non sono tutte canoniche. Le coperture canoniche corrispondenti sono:

a) $X = ABC$: $\{AB \to C, AC \to B, BC \to A\}$;
b) $X = ABCD$: $\{B \to D, AB \to C, AC \to B, BC \to A\}$;
c) $X = ABCEG$: $\{E \to G, AB \to C, AC \to B, BC \to A, BC \to E\}$;
d) $X = ABCH$: $\{AB \to C, AC \to B, BC \to A\}$;
e) $X = ABCDEGH$: $\{B \to D, E \to G, AB \to C, AC \to B, BC \to A, AD \to E\}$.

Osserviamo che, se $Y_A \to A$ è una dipendenza non banale in F^+, allora $BC \subseteq Y_A$, ed analogamente $Y_B \to B$ e $Y_C \to C$ implicano $AC \subseteq Y_B$ e $AB \subseteq Y_C$, per cui ogni chiave di uno $X \supseteq ABC$ deve includere AB oppure AC oppure BC. D'altra parte, la chiusura di AB, AC, e BC, contiene tutti gli attributi tranne H. Da questo possiamo dedurre l'insieme delle chiavi e degli attributi primi dei cinque schemi.

a) $X = ABC$: $\{AB \to C, AC \to B, BC \to A\}$; chiavi: $\{AB, AC, BC\}$; primi: $\{ABC\}$.
b) $X = ABCD$: $\{B \to D, AB \to C, AC \to B, BC \to A\}$; chiavi: $\{AB, AC, BC\}$; primi: $\{ABC\}$.
c) $X = ABCEG$: $\{E \to G, AB \to C, AC \to B, BC \to A, BC \to E\}$; chiavi: $\{AB, AC, BC\}$; primi: $\{ABC\}$.
d) $X = ABCH$: $\{AB \to C, AC \to B, BC \to A\}$; chiavi: $\{ABH, ACH, BCH\}$; primi: $\{ABCH\}$.
e) $X = ABCDEGH$: $\{B \to D, E \to G, AB \to C, AC \to B, BC \to A, AD \to E\}$; chiavi: $\{ABH, ACH, BCH\}$; primi: $\{ABCH\}$.

A questo punto è facile indicare per ogni schema la sua forma normale.

a) $X = ABC$: $\{AB \to C, AC \to B, BC \to A\}$; chiavi: $\{AB, AC, BC\}$; primi: $\{ABC\}$, BCNF e quindi 3NF.
b) $X = ABCD$: $\{B \to D, AB \to C, AC \to B, BC \to A\}$; chiavi: $\{AB, AC, BC\}$; primi: $\{ABC\}$, $B \to D$ viola le due forme normali.

c) $X = ABCEG$: $\{E \to G, AB \to C, AC \to B, BC \to A, BC \to E\}$; chiavi: $\{AB, AC, BC\}$; primi: $\{ABC\}$, $E \to G$ viola le due forme normali.
d) $X = ABCH$: $\{AB \to C, AC \to B, BC \to A\}$; chiavi: $\{ABH, ACH, BCH\}$; primi: $\{ABCH\}$, $AB \to C$ viola le due forme normali.
e) $X = ABCDEGH$: $\{B \to D, E \to G, AB \to C, AC \to B, BC \to A, AD \to E\}$; chiavi: $\{ABH, ACH, BCH\}$; primi: $\{ABCH\}$, $B \to D$ viola le due forme normali.

Esercizio 5.7 Si consideri la relazione con schema $R(A, B, C, D)$ e si supponga che l'unica istanza possibile di R sia:

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d2
a2	b1	c1	d3
a2	b1	c3	d4

Si trovi una copertura canonica delle dipendenze funzionali soddisfatte da R.

Soluzione 5.7
Consideriamo per prima cosa le dipendenze con a sinistra un singoletto. Abbiamo: $A \to B$, $C \to B$, $D \to ABC$. Tra i determinanti con due attributi, è inutile considerare quelli che contengono D, poiché D è chiave da solo, né quelli che contengono B, poiché B è costante. Rimane la dipendenza $AC \to BD$. Portando in forma canonica, otteniamo: $\{A \to B, C \to B, D \to A, D \to C, AC \to D\}$.

Esercizio 5.8 Si consideri il seguente schema relazionale:

Impiegati(Nome, Livello, Stipendio)

per il quale valgono le seguenti dipendenze funzionali
Nome → Livello Stipendio
Livello → Stipendio

a) Lo schema è in 3NF o in BCNF?
b) Se lo schema non è in 3NF, si applichi l'algoritmo di sintesi per trovare una decomposizione che preservi dati e dipendenze.
c) Se lo schema non è in BCNF, si applichi l'algoritmo di decomposizione per trovare uno decomposizione in BCNF e dire se conserva le dipendenze.

Soluzione 5.8
Abbreviamo lo schema come segue: $I(N, L, S), \{N \to LS, L \to S\}$. Portandolo in forma canonica, abbiamo: $\{N \to L, L \to S\}$, chiavi: $\{N\}$.

a) Lo schema non è in 3NF né, quindi, in BCNF, per la dipendenza $L \rightarrow S$.
b) Algoritmo di sintesi: $R1(NL), \{N \rightarrow L\}; R2(LS), \{L \rightarrow S\}$. Non c'è bisogno di aggiungere schemi poiché NL è già una superchiave.
c) Algoritmo di analisi: poiché $L \rightarrow S$ viola la 4NF, decompongo in $R1(LS), \{L \rightarrow S\}$ ed $R1(LN), \{N \rightarrow L\}$, i due schemi sono in 4NF, per cui ho finito. La decomposizione ha preservato le dipendenze.

Esercizio 5.9 Si supponga che per archiviare dati sull'inventario delle apparecchiature di un'azienda sia stata usata una tabella con la seguente struttura:

Inventario(NInventario, Modello, Descrizione, NSerie, Costo, Responsabile, Telefono)

NInventario	Modello	Descrizione	NSerie	Costo
...	...	...	...	...
111	SUN3	Stazione Sun	ajk0785	25 000
112	PB180	Notebook Mac	a908m	6000
113	SUN3	Stazione Sun	ajp8907	27 000
...	...	...	...	...

Responsabile	Telefono
...	...
Caio	576
Tizio	587
Tizio	587
...	...

Il numero di inventario identifica un'apparecchiatura. Un'apparecchiatura ha un costo, un modello e un numero di serie. Apparecchiature dello stesso modello possono avere costi differenti, perché acquistate in momenti diversi, ma hanno la stessa descrizione. Il numero di serie è una caratteristica dell'apparecchiatura e due diverse apparecchiature dello stesso modello hanno numero di serie diverso. Ogni apparecchiatura ha un responsabile, che può avere più apparecchiature, ma un unico numero di telefono. I responsabili sono identificati dal cognome.

Definire le dipendenze funzionali e dire se lo schema proposto presenta anomalie, giustificando la risposta.

Trasformare la rappresentazione in uno schema relazionale in 3NF.

Soluzione 5.9
Abbreviamo lo schema come segue: *Inv(NInv, Mod, Descr, NSerie, Costo, Resp, Tel)*.
Dal testo ricaviamo le seguenti dipendenze:

a) NInv $\rightarrow$ Mod NSerie Costo

b) Mod → Descr
c) Mod NSerie → NInv
d) NInv → Resp
e) Resp → Tel

Le chiavi dello schema sono {NInv; (ModNSerie)}. Lo schema presenta numerose anomalie, testimoniate dalle dipendenze (2), (4) e (5). Applicando l'algoritmo di sintesi, otteniamo il seguente schema, che rispetta sia la 3NF che la BCNF:
Apparecchiature(NInv, Mod, NSerie, Costo, Resp),
Modelli(Mod, Descr),
Responsabili(Resp, Tel).

Esercizio 5.10 Una palestra ospita diversi corsi appartenenti a diverse tipologie (aerobica, danza moderna, ...). Ogni corso ha una sigla, che lo identifica, un insegnante e alcuni allievi. Un insegnante offre in generale più corsi, anche con diverse tipologie, e anche un allievo può essere iscritto a più corsi. Di ogni insegnante interessano il nome (che lo identifica) e l'indirizzo. Di ogni allievo interessano il nome (che lo identifica) e il numero di telefono. Per ogni allievo interessa sapere, per ogni corso che frequenta, quanto ha già versato finora. La palestra gestisce attualmente i dati con un foglio elettronico con tante colonne quanti sono i fatti elementari da trattare.
Si chiede di:

a) Definire le dipendenze funzionali.
b) Dare una copertura canonica delle dipendenze in tale schema ed elencare le chiavi.
c) Applicare allo schema l'algoritmo di sintesi per portarlo in 3NF, e dire se lo schema così ottenuto è anche in BCNF.
d) Applicare allo schema l'algoritmo di decomposizione per portarlo in BCNF, e dire se tale decomposizione preserva le dipendenze.

Soluzione 5.10
Le colonne dello schema attuale sono le seguenti, di cui riportiamo un'abbreviazione: SiglaCorso (SiglaC), Tipologia (Tipo), NomeInsegnante (NomeI), IndirizzoInsegnante (IndI), NomeAllievo (NomeA), TelefonoAllievo (TelA), VersatoFinora (Vers).

a) Dipendenze funzionali:
   - SiglaC → Tipo NomeI
   - NomeI → IndI
   - NomeA → TelA
   - NomeA SiglaC → Vers

Non è chiaro se si assume anche "TelA $\rightarrow$ NomeA", ma la cosa non fa molta differenza (se non per ciò che riguarda l'insieme delle chiavi).

b) Le dipendenza sono già in forma minima. La sola chiave è la coppia *(NomeA, SiglaC)*.

c) Applicando l'algoritmo di sintesi otteniamo lo schema:
Corsi(SiglaC, Tipo, NomeI),
Insegnanti(NomeI, indI),
Allievi(NomeA, TelA),
SoldiVersati(NomeA, SiglaC, Vers),
che è anche in BCNF.

d) Considerando le dipendenze nell'ordine in cui sono elencate, otteniamo la seguente decomposizione:
R(SiglaC, Tipo, NomeI, IndI, NomeA, TelA, Vers),
$\{SiglaC \rightarrow Tipo\ NomeI, NomeI \rightarrow IndI, NomeA \rightarrow TelA, NomeA\ SiglaC \rightarrow Vers\} \Rightarrow$
R1(SiglaC, Tipo, NomeI, IndI), $\{SiglaC \rightarrow Tipo\ NomeI, NomeI \rightarrow IndI\}$,
R2(SiglaC, NomeA, TelA, Vers), $\{NomeA \rightarrow TelA, NomeA\ SiglaC \rightarrow Vers\} \Rightarrow$
R11(NomeI, IndI), $\{NomeI \rightarrow IndI\}$,
R12(NomeI, SiglaC Tipo), $\{SiglaC \rightarrow Tipo\ NomeI\}$,
R2(SiglaC, NomeA, TelA, Vers), $\{NomeA \rightarrow TelA, NomeA\ SiglaC \rightarrow Vers\} \Rightarrow$
R11(NomeI, IndI), $\{NomeI \rightarrow IndI\}$,
R12(NomeI, SiglaC, Tipo), $\{SiglaC \rightarrow Tipo\ NomeI\}$,
R21(NomeA, TelA), $\{NomeA \rightarrow TelA\}$,
R22(NomeA, SiglaC, Vers), $\{NomeA\ SiglaC \rightarrow Vers\}$.
Questa decomposizione è uguale a quella ottenuta usando l'algorimo di sintesi, e conserva anche le dipendenze.

Esercizio 5.11 Quali di questi test ammettono algoritmi di complessità polinomiale:

a) Dato lo schema di relazione $R\langle T, F\rangle$, $A \in T$, A è primo?
b) Dati due insiemi di dipendenze F e G, $F \equiv G$?
c) Dato lo schema di relazione $R\langle T, F\rangle$, e $X \subseteq T$, X è una superchiave?
d) Dato lo schema di relazione $R\langle T, F\rangle$, e $X \subseteq T$, X è una chiave?

Soluzione 5.11

a) *Problema*: dato lo schema di relazione $R\langle T, F\rangle$, $A \in T$, A è primo?
Discussione: Un algoritmo per valutare la primalità di un attributo A consiste nel generare tutti i sottoinsiemi di T che contengono A e verificare se uno

di essi sia una chiave. Questo algoritmo ha complessità esponenziale rispetto al numero di attributi, poiché il numero di sottoinsiemi di un insieme di a elementi è 2^a, e verificare se un sottoinsieme di un insieme di a attributi sia una chiave, rispetto ad un insieme di p dipendenze, ha complessità polinomiale $O(ap)$. Non sono noti algoritmi polinomiali.

b) *Problema*: Dati due insiemi di dipendenze F e G, $F \equiv G$?
Discussione: Il problema ammette un algoritmo polinomiale. È sufficiente verificare, per ogni $X \to Y \in F$ che si abbia $Y \subseteq X_G^+$, e che per ogni $X \to Y \in G$ si abbia che $Y \subseteq X_F^+$. Dato che la chiusura e l'inclusione possono essere valutate in tempo $O(ap)$, la complessità dell'algoritmo è quindi $O(ap^2)$.

c) *Problema*: Dato lo schema di relazione $R\langle T, F\rangle$, e $X \subseteq T$, X è una superchiave?
Discussione: È sufficiente verificare se $T = X_F^+$, con complessità $O(ap)$.

d) *Problema*: Dato lo schema di relazione $R\langle T, F\rangle$, e $X \subseteq T$, X è una chiave?
Discussione: È sufficiente verificare che si abbia $X_F^+ = T$ e che, per ogni $A \in X$, si abbia $(X - A)_F^+ \neq T$. La complessità è quindi $O(a^2p)$.

Esercizio 5.12 Dato lo schema $R\langle T, F\rangle$, discutere la complessità dei seguenti problemi:

a) Trovare una chiave di R.
b) Trovare tutte le chiavi di R.
c) $F - \{X \to Y\} \equiv F$.

Soluzione 5.12

a) Trovare una chiave di $R\langle T, F\rangle$: si può utilizzare il seguente algoritmo, dove `A[1..n]` enumera gli attributi in T:

```
input: A[1..n], F;
K := [A[1],...,A[n]];
for i in 1..n
    do if chiudi((K-A[i]),F) = T then K := K-A[i];
return(K)
```

È facile dimostrare le seguenti invarianti:

- sia K_i il valore di K dopo il ciclo i-esimo; per ogni i, si ha che $K_i \to T$;
- per ogni $j \leqslant i$, se l'attributo $A[j]$ appartiene a K_i, allora $A[j]$ non è estraneo in $K_i \to T$.

Ponendo $i = n$, abbiamo quindi che K_n è una superchiave senza attributi estranei, ovvero è una chiave.

b) Trovare tutte le chiavi di $R\langle T, F\rangle$: in generale, il numero di chiavi di uno schema può crescere in modo esponenziale con le dimensioni dello schema stesso, per cui nessun algoritmo che le enumeri può avere una complessità meno che esponenziale rispetto alle dimensioni dello schema.

c) $F - \{X \to Y\} \equiv F$: basta verificare se $X^+_{F-\{X\to Y\}} \supseteq Y$, con costo $O(ap)$.

Esercizio 5.13 Discutere la complessità dei seguenti test:

a) *TEST 3NF*: dato lo schema di relazione $R\langle T, F\rangle$, R è in 3NF rispetto ad F?
b) *TEST BCNF*: dato lo schema di relazione $R\langle T, F\rangle$, R è in BCNF rispetto ad F?
c) *TEST BCNF DI SOTTOSCHEMA*: dato lo schema di relazione $R\langle T, F\rangle$, dato $X \subseteq T$, X è in BCNF rispetto alla proiezione di F su X?
d) *TEST COPERTURA CANONICA*: dato lo schema di relazione $R\langle T, F\rangle$, F è in forma canonica?

Soluzione 5.13

a) *TEST 3NF*. Un modo per determinare se uno schema $R\langle T, F\rangle$ sia in 3NF consiste nel portare lo schema in una forma canonica G e poi nel verificare se, per ogni dipendenza in $X \to A$ in G, se X non è una chiave, allora A è primo. Per portare F in forma canonica è sufficiente:
   - dividere le dipendenze in modo che ogni membro destro sia composto di un solo attributo;
   - eliminare, da ogni membro sinistro, gli attributi ridondanti;
   - eliminare le dipendenze ridondanti.

   Tutte queste operazioni si possono effettuare in tempo polinomiale.
   Data poi una dipendenza $X \to A$ in G, verificare se X sia chiave si può fare in tempo polinomiale, poiché significa verificare se $X^+ = T$; tuttavia, quando X non è chiave, verificare se A sia primo usando l'algoritmo sopra descritto richiede un tempo esponenziale. Quindi questo algoritmo ha complessità esponenziale. Non sono noti algoritmi polinomiali.

b) *TEST BCNF*. Questo problema ha complessità polinomiale; basta utilizzare lo stesso algoritmo visto al punto precedente per portare F in forma canonica G. Per ogni $X \to A$ in G bisogna poi verificare se X sia una chiave, e questa operazione è polinomiale.

Proiezione delle dipendenze: per ogni $Y \subset X$ non vuoto si calcoli Y_F^+, e si generi in questo modo l'insieme $G = \{Y \to A \mid Y \subset X,\ A \in ((Y_F^+ - Y) \cap X)\}$. Questa operazione ha una complessità esponenziale rispetto alla dimensione di X. Si porti poi G in forma canonica, con un costo polinomiale rispetto alla dimensione di G (che, nel caso pessimo, è esponenziale rispetto alla dimensione di X).

In pratica, si possono adottare gli accorgimenti descritti in 5.6 per ridurre la quantità di sottoinsiemi di X da considerare, ma questi non bastano a rendere la complessità di questo algoritmo meno che esponenziale. Non sono noti algoritmi polinomiali.

c) *TEST BCNF DI SOTTOSCHEMA*. Per risolvere questo problema si può operare come segue: prima si proietta F su X, operando come descritto sopra, e poi si verifica se $R\langle X, \Pi_X F\rangle$ è in BCNF. Questo algoritmo è esponenziale, dato che questo è il costo della fase di proiezione.

d) *TEST COPERTURA CANONICA*. L'algoritmo più semplice verifica le seguenti condizioni:

- Tutti i membri destri sono formati da un solo attributo: costo $O(p)$
- Per ogni $X \to A \in F$, per ogni $B \in X$, $A \notin (X - B)^+$: costo $p \times a \times O(ap) = O(a^2p^2)$
- Per ogni $X \to A \in F$, $A \in X^+$, dove la chiusura è calcolata rispetto ad $F - \{X \to A\}$: costo $p \times O(ap) = O(ap^2)$

Costo totale dell'algoritmo: $O(a^2p^2)$.

Esercizio 5.14 Si supponga che una dipendenza funzionale $X \to Y$ sia soddisfatta da un'istanza di relazione r. Sia $s \subseteq r$ (quindi s è una relazione con gli stessi attributi di r). s soddisfa $X \to Y$? Se sì, dire perché, altrimenti dare un controesempio.

Soluzione 5.14
Se un'istanza di relazione r soddisfa una dipendenza $X \to Y$, allora $\forall t_1, t_2 \in r$. $t_1[X] = t_2[x] \to t_1[Y] = t_2[Y]$. Da $s \subseteq r$ segue quindi che anche $\forall t_1, t_2 \in s$. $t_1[X] = t_2[x] \to t_1[Y] = t_2[Y]$, per cui s soddisfa $X \to Y$.

Esercizio 5.15 Si supponga che una dipendenza funzionale $X \to Y$ sia soddisfatta da due istanze di relazione r ed s con gli stessi attributi.
$r \cap s$ soddisfa $X \to Y$? $r \cup s$ soddisfa $X \to Y$? Se sì, dire perché, altrimenti dare un controesempio.

Soluzione 5.15
Ragionando come in 5.14, si dimostra che $r \cap s$ soddisfa $X \to Y$. Invece $r \cup s$

potrebbe non soddisfare $X \to Y$. Si considerino le istanze di relazione $\{(A = 1, B = 1)\}$ e $\{(A = 1, B = 2)\}$, con schema $R(AB)$. Ciascuna delle due soddisfa $A \to B$, tuttavia la loro unione $\{(A = 1, B = 1), (A = 1, B = 2)\}$ non soddisfa la dipendenza.

Esercizio 5.16
Sia $R\langle T, F\rangle$ uno schema relazionale, con F una copertura canonica. Si dimostri che se lo schema ha una sola chiave ed è in 3NF allora è anche in BCNF. (Suggerimento: si inizi con lo scrivere la definizione di 3NF e di BCNF, e si dia un nome, ad esempio Y, all'insieme di attributi che forma la sola chiave di $R\langle T, F\rangle$. Si ragioni per assurdo, supponendo che $R\langle T, F\rangle$ sia in 3NF ma non in BCNF).

Soluzione 5.16
Sia Y la sola chiave di $R\langle T, F\rangle$, e assumiamo, per assurdo, che $R\langle T, F\rangle$ sia in 3NF ma non in BCNF. Ne segue che esiste una dipendenza non banale $X \to A$ tale che X non è superchiave ma $A \in Y$. Da $X \to A$ segue che $XY - A$ determina Y, per cui $XY - A$ è una superchiave, per cui contiene una chiave la quale non contiene A. Quindi $R\langle T, F\rangle$ ha almeno due chiavi, contraddicendo l'ipotesi.

Esercizio 5.17 Dimostrare il Teorema 5.9:
Uno schema $R\langle T, F\rangle$ è in BCNF se e solo se per ogni dipendenza funzionale non banale $X \to Y \in F$, X è una superchiave.

Soluzione 5.17
Dato uno schema $R\langle T, F\rangle$ vogliamo dimostrare che, se per ogni $X \to Y \in F$ non banale X è una superchiave, allora per ogni $X \to Y \in F^+$ non banale X è una superchiave (l'implicazione inversa è immediata dato che $F \subseteq F^+$). A tale scopo basta osservare che F^+ deriva da F tramite l'applicazione degli assiomi di Armostrong, e questi, partendo da uno schema che soddisfa la condizione sopra specificata, producono ancora solo dipendenze che o sono banali o hanno a sinistra una superchiave. Infatti, la riflessività aggiunge solo dipendenze banali. L'arricchimento ricava una dipendenza con determinante XW a partire da una con determinante X; se la dipendenza ottenuta è non banale allora anche quella originale era non banale, per cui X è una superchiave, per cui XW è una superchiave. Si consideri infine la derivazione di $X \to Z$ a partire da $X \to Y$ e $Y \to Z$ per transitività. Se $X \to Y$ e $Y \to Z$ sono entrambe banali, allora anche $X \to Z$ è banale. Se almeno una è non banale allora almeno una tra X ed Y è superchiave, e quindi, dato che X determina Y, X è superchiave.

Esercizio 5.18 Dimostrare il Teorema 5.11:
Uno schema $R\langle T, F\rangle$ è in 3NF se e solo se, per ogni dipendenza funzionale $X \to A_1, \ldots, A_n \in F$, e per ogni $i \in \{1 \ldots n\}$, $A_i \in X$ oppure X è una superchiave oppure A_i è primo.

Soluzione 5.18

Diciamo che uno schema $R\langle T, F\rangle$ soddisfa la condizione (p) se per ogni $X \to A_1 \dots A_n \in F$ e per ogni i, si ha che o vale X superchiave (p^1_X) o vale $A_i \in X$ $(p^2_{A_i,X})$, o vale A_i *primo* $(p^3_{A_i})$. Nel seguito, abbreviamo spesso (p^1_X), $(p^2_{A_i,X})$ e $(p^3_{A_i})$ a (p^1), (p^2) e (p^3). Vogliamo dimostrare che, se se $R\langle T, F\rangle$ soddisfa C, allora $R\langle T, F\rangle$ è in 3NF (l'implicazione inversa è banale).

A tale scopo basta osservare che F^+ deriva da F tramite l'applicazione degli assiomi di Armostrong, e questi, partendo da uno schema che soddisfa (p), producono ancora solo dipendenze che soddisfano (p). Infatti, la riflessività aggiunge solo dipendenze che soddisfano (p^2). L'arricchimento ricava $XW \to A_1 \dots A_n W$ a partire da $X \to A_1 \dots A_n$; se valeva (p^1_X), allora vale (p^1_{XW}) per la nuova dipendenza. Altrimenti, ciascuno degli attributi A_i continua a soddisfare (p^2) o (p^3) come in $X \to A_1 \dots A_n$, e tutti gli attributi in W soddisfano (p^2), dato che appartengono a XW.

Si consideri infine la derivazione di $X \to A_1 \dots A_n$ a partire da $X \to Y$ e $Y \to A_1 \dots A_n$ per transitività, e si consideri un A_i. Se X è superchiave, allora (p^1_X) vale, e abbiamo finito. Se invece X non è superchiave, allora neppure Y è superchiave, e quindi, dato che $Y \to A_i$ soddisfa (p), o A_i soddisfa $(p^3_{A_i})$, oppure A_i non soddisfa $(p^3_{A_i})$ e $A_i \in Y$. In quest'ultimo caso, dato che $X \to Y$ soddisfa (p), X non è superchiave, A_i non soddisfa (p^3), e $A_i \in Y$, allora A_i deve soddisfare $(p^2_{A_i,X})$.

Capitolo 6

SQL PER L'USO INTERATTIVO DI BASI DI DATI

Esercizio 6.1 Dare un'espressione **SELECT** per stabilire se i valori di A in una relazione con schema R(A, B, C) siano tutti diversi. L'espressione deve essere diversa da **SELECT** A **FROM** R.

Soluzione 6.1
La seguente espressione restituisce true se i valori di A sono tutti diversi, false altrimenti.

```
SELECT    COUNT(*) = COUNT(DISTINCT A)
FROM      R;
```

Si noti che per stabilire se un insieme di attributi {B, C} determina l'attributo A, si controlla se la seguente interrogazione ritorna una tabella vuota:

```
SELECT    B, C, COUNT(DISTINCT A)
FROM      R
GROUP BY  B, C
HAVING    COUNT(DISTINCT A) > 1;
```

Esercizio 6.2 Si ricorda che il predicato **IN** è equivalente a =**ANY**. Spiegare perché il predicato **NOT IN** non è equivalente a <>**ANY** ma a <>**ALL**.

Soluzione 6.2 Da Fare.

Esercizio 6.3 È importante sapere quando una **SELECT** ritorna una tabella con righe diverse per evitare di usare inutilmente la clausola **DISTINCT**, che comporta un costo addizionale per l'esecuzione dell'interrogazione (perché?).

a) È vero che con una **SELECT** con una tabella nella parte **FROM**, e senza **GROUP BY**, non vi saranno righe duplicate nel risultato se gli attributi della parte **SELECT** sono una superchiave della tabella?
b) È vero che con una **SELECT** con più tabelle nella parte **FROM**, e senza **GROUP BY**, non vi saranno righe duplicate nel risultato se gli attributi della parte **SELECT** sono una superchiave di ogni tabella?
c) È vero che nessuna interrogazione con un **GROUP BY** può avere duplicati nel risultato?

Per ogni caso dare un esempio di interrogazione che ritorna duplicati se la condizione non è verificata.

Soluzione 6.3 Vedi testo.

Esercizio 6.4 Si consideri lo schema relazionale

Studenti(Matricola, Nome, Provincia, AnnoNascita)
Esami(Materia, MatricolaStudente, Voto, NumAppello, Anno)

Formulare in SQL le seguenti interrogazioni:

a) Trovare il nome degli studenti di Pisa che hanno superato l'esame di Programmazione con 30.
b) Trovare il nome degli studenti che hanno superato cinque esami.
c) Trovare per ogni studente di Pisa il numero degli esami superati, il voto massimo, minimo e medio.
d) Trovare per ogni materia il numero degli esami fatti al primo appello, il voto massimo, minimo e medio.

Soluzione 6.4
Di solito un'interrogazione può essere espressa in SQL in modi diversi. Solo in qualche caso si mostrano più soluzioni.

a)
```
SELECT    Nome
FROM      Studenti, Esami
WHERE     Matricola = MatricolaStudente AND Provincia = 'Pisa'
AND       Materia = 'Programmazione' AND Voto = 30;
```

b)
```
SELECT      Nome
FROM        Studenti, Esami
WHERE       MatricolaStudente = Matricola
GROUP BY    Nome
HAVING      COUNT(*) = 5;
```
oppure:
```
SELECT      Nome
FROM        Studenti
WHERE       5 = ( SELECT  COUNT(*)
                  FROM    Esami
                  WHERE   MatricolaStudente = Matricola);
```

Questa soluzione è meno efficiente della precedente perché la sottoselect viene calcolata per ogni record della tabella Studenti.

c)
```
SELECT      Matricola, Nome, COUNT(*) AS EsamiSuperati,
            MAX(Voto), MIN(Voto), AVG(Voto)
FROM        Studenti, Esami
WHERE       Matricola = MatricolaStudente AND Provincia = 'Pisa'
GROUP BY    Matricola, Nome;
```

d)
```
SELECT      Materia, COUNT(*) AS NumeroEsami,
            MAX(Voto), MIN(Voto), AVG(Voto)
FROM        Esami
WHERE       NumAppello = 1
GROUP BY    Materia;
```

Esercizio 6.5 Usando la base di dati relazionale ottenuta dall'Esercizio 2.2.9, formulare in SQL le seguenti interrogazioni:

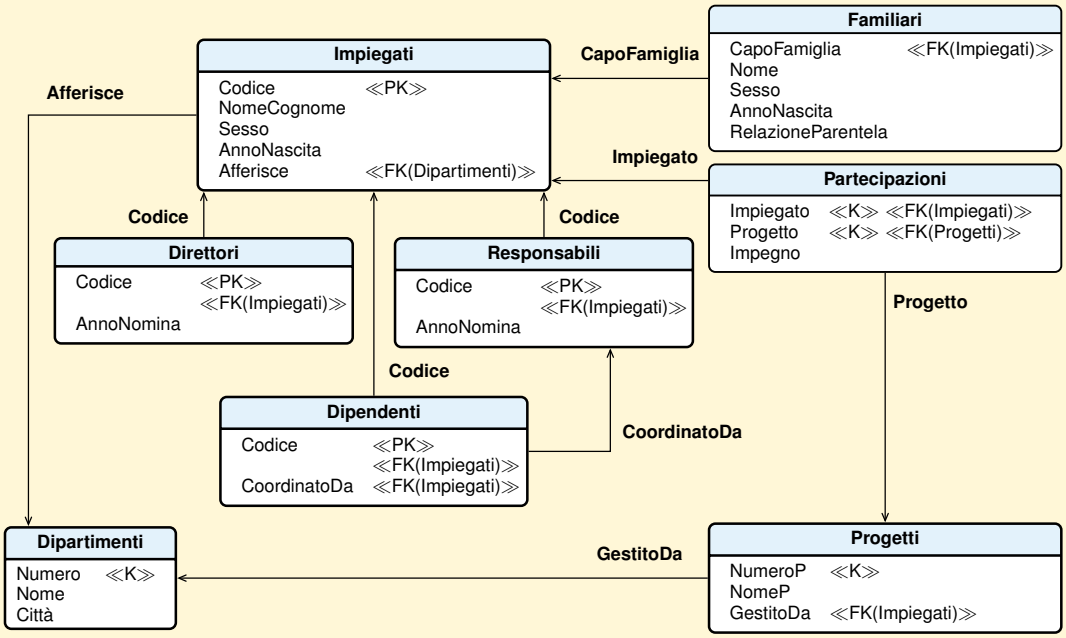
a) Trovare il nome e l'anno di nascita dei figli dell'impiegato con codice 350.
b) Trovare il nome e codice degli impiegati e il nome del dipartimento dove lavorano.
c) Trovare il nome degli impiegati, il nome e l'anno di nascita dei figli maschi a carico.
d) Trovare, per ogni progetto in corso a Pisa, il numero e il nome del progetto, il nome del dipartimento dove si svolge, il cognome del direttore del dipartimento.
e) Trovare il nome dei dipartimenti con almeno un impiegato con persone a carico.
f) Trovare il numero degli impiegati del dipartimento di Informatica.
g) Trovare, per ogni progetto al quale lavorano più di due impiegati, il nome, il numero e il numero degli impiegati che vi lavorano.
h) Trovare per ogni dipartimento il nome, il numero degli impiegati e la media del loro anno di nascita.
i) Trovare i nomi dei supervisori e dei loro dipendenti.
j) Trovare il nome degli impiegati e il nome del dipartimento in cui lavorano.
k) Trovare il nome degli impiegati senza familiari a carico.

l) Trovare i progetti cui partecipa il sig. Rossi come impiegato o come direttore del dipartimento che gestisce il progetto.
m) Trovare il nome degli impiegati che hanno i familiari a carico dello stesso sesso.
n) Trovare il nome degli impiegati che hanno tutti i familiari a carico dello stesso sesso dell'impiegato.
o) Trovare il nome degli impiegati che lavorano almeno a tutti i progetti dell'impiegato con codice 300.
p) Trovare il nome del dipartimento e il numero di impiegati nati dopo il 1950, per i dipartimenti con più di due impiegati.

Soluzione 6.5

Anche per questo esercizio solo in alcuni casi si mostrano più soluzioni.

noindent Lo schema relazionale è mostrato in figura. Per poter provare queste ed altre interrogazioni nel sistema JRS sono disponibili nel file http://fondamentidibasidati.it/AziendaPerJRS.zip i comandi per la creazione della base di dati, una serie di comandi per inserire dei dati e le interrogazioni.



Tutte le query assumono l'esistenza delle seguenti viste:

```
CREATE    VIEW DatiDirettori
          (Codice, NomeCognome, Sesso, AnnoNascita, Afferisce,
          Dirige, AnnoNomina)
          AS
SELECT    i.Codice, i.NomeCognome, i.Sesso, i.AnnoNascita,
          i.Afferisce, d.Dirige, d.AnnoDiNomina
FROM      Impiegati i, Direttori d
WHERE     i.Codice = d.Codice;

CREATE    VIEW DatiResponsabili
          (Codice, NomeCognome, Sesso, AnnoNascita, Afferisce,
          AnnoNomina)
          AS
SELECT    i.Codice, i.NomeCognome, i.Sesso, i.AnnoNascita,
          i.Afferisce, r.AnnoDiNomina
FROM      Impiegati i, Responsabili r
WHERE     i.Codice = r.Codice;

CREATE    VIEW DatiDipendenti
          (Codice, NomeCognome, Sesso, AnnoNascita, Afferisce,
          CoordinatoDa)
          AS
SELECT    i.Codice, i.NomeCognome, i.Sesso, i.AnnoNascita,
          i.Afferisce, d.CoordinatoDa
FROM      Impiegati i, Dipendenti d
WHERE     i.Codice = d.Codice;
```

a)
```
SELECT    Nome, AnnoNascita
FROM      Familiari
WHERE     CapoFamiglia = 350 AND RelazioneParentela = 'figlio';
```

b)
```
SELECT    i.NomeCognome, i.Codice, d.Nome
FROM      Impiegati i, Dipartimenti d
WHERE     i.Afferisce = d.Numero;
```

c)
```
SELECT    i.NomeCognome, f.Nome, f.AnnoNascita
FROM      Impiegati i, Familiari f
WHERE     f.CapoFamiglia = i.Codice
AND       f.RelazioneParentela = 'figlio' AND f.Sesso = 'm';
```

d)
```
SELECT    p.NumeroP, p.NomeP, d.Nome, dd.NomeCognome
FROM      Progetti p, Dipartimenti d, DatiDirettori dd
WHERE     p.GestitoDa = d.Numero AND dd.Dirige = d.Numero
AND       d.Citta = 'Pisa';
```

e)
```
SELECT    DISTINCT d.Nome
FROM      Dipartimenti d, Impiegati i, Familiari f
WHERE     i.Afferisce = d.Numero AND f.CapoFamiglia = i.Codice;
```
oppure:
```
SELECT    d.Nome
FROM      Dipartimenti d
WHERE     EXISTS(SELECT *
                 FROM   Impiegati i
                 WHERE  i.Afferisce=d.Numero
                        AND EXISTS(SELECT *
                                   FROM   Familiari f
                                   WHERE  f.CapoFamiglia=i.Codice ));
```

f)
```
SELECT    COUNT(*)
FROM      Dipartimenti d, Impiegati i
WHERE     d.Nome = 'Informatica' AND d.Numero = i.Afferisce;
```

g)
```
SELECT    p.NomeP, p.NumeroP, COUNT(*) AS NumeroImpiegati
FROM      Progetti p, Impiegati i, Partecipazioni pa
WHERE     p.NumeroP = pa.Progetto AND pa.Impiegato = i.Codice
GROUP BY  p.NumeroP, p.NomeP HAVING COUNT(*)>2;
```

h)
```
SELECT    d.Nome, COUNT(*) AS NumeroImpiegati,
          AVG(i.AnnoNascita) AS MediaAnnoNascImpiegati
FROM      Dipartimenti d, Impiegati i
WHERE     d.Numero=i.Afferisce
GROUP BY  d.Numero, d.Nome;
```

i)
```
SELECT    dr.NomeCognome, ddi.NomeCognome
FROM      DatiResponsabili dr, DatiDipendenti ddi
WHERE     ddi.CoordinatoDa = dr.Codice;
```

j)
```
SELECT    i.NomeCognome, d.Nome
FROM      Impiegati i, Dipartimenti d
WHERE     i.Afferisce = d.Numero;
```

k)
```
SELECT    i.NomeCognome
FROM      Impiegati i
WHERE     NOT EXISTS(SELECT *
                     FROM   Familiari f
                     WHERE  f.CapoFamiglia = i.Codice);
```
oppure:

```
SELECT    NomeCognome
FROM      Impiegati
  EXCEPT
SELECT    NomeCognome
FROM      Impiegati , Familiari
WHERE     Codice = CapoFamiglia;
```

l)
```
SELECT    p.NomeP, i.Codice, i.NomeCognome
FROM      Progetti p, Partecipazioni pa, Impiegati i, DatiDirettori dd,
          Dipartimenti d
WHERE     p.NumeroP = pa.Progetto AND pa.Impiegato = i.Codice
AND       p.GestitoDa = d.Numero AND dd.Dirige = d.Numero
AND       (i.NomeCognome = 'Pablo Rossi' OR
          dd.NomeCognome = 'Pablo Rossi');
```

oppure:

```
SELECT    p.NomeP, i.Codice AS c, i.NomeCognome AS n
FROM      Progetti p, Partecipazioni pa, Impiegati i
WHERE     p.NumeroP = pa.Progetto AND pa.Impiegato = i.Codice
AND       i.NomeCognome = 'Pablo Rossi'
  UNION
SELECT    p.NomeP, dd.Codice AS c, dd.NomeCognome AS n
FROM      Progetti p, DatiDirettori dd, Dipartimenti d
WHERE     p.GestitoDa = d.Numero AND dd.Dirige = d.Numero
AND       dd.NomeCognome = 'Pablo Rossi';
```

m) La soluzione con il quantificatore universale è:

```
SELECT    i.NomeCognome
FROM      Impiegati i
WHERE     (FOR ALL f IN Familiari WHERE f.CapoFamiglia = i.Codice:
              f.Sesso = 'm') OR
          (FOR ALL f IN Familiari WHERE f.CapoFamiglia = i.Codice:
              f.Sesso = 'f');
```

con la doppia negazione si ottiene:

```
SELECT    i.NomeCognome
FROM      Impiegati i
WHERE     (FOR SOME f IN Familiari WHERE f.CapoFamiglia = i.Codice:
              f.Sesso = 'm') AND
          (FOR SOME f IN Familiari WHERE f.CapoFamiglia = i.Codice:
              f.Sesso = 'f');
```

che in SQL diventa:

```
SELECT    i.NomeCognome
FROM      Impiegati i
WHERE     EXISTS(SELECT *
                 FROM   Familiari f
                 WHERE  f.CapoFamiglia = i.Codice AND f.Sesso = 'm')
AND       EXISTS(SELECT *
                 FROM   Familiari f
                 WHERE  f.CapoFamiglia = i.Codice AND f.Sesso = 'f');
```

L'interrogazione si può anche scrivere senza sottoselect, utilizzando opportunamente le giunzioni e le funzioni di aggregazione:

```
SELECT    i.NomeCognome
FROM      Impiegati i, Familiari f
WHERE     f.CapoFamiglia = i.Codice
GROUP BY  i.Codice, i.NomeCognome
HAVING    COUNT(DISTINCT f.Sesso) = 1;
```

n) La soluzione con il quantificatore universale è:

```
SELECT    i.NomeCognome
FROM      Impiegati i
WHERE     FOR ALL f IN Familiari WHERE f.CapoFamiglia = i.Codice:
              f.Sesso = i.Sesso;
```

con la doppia negazione si ottiene:

```
SELECT    i.NomeCognome
FROM      Impiegati i
WHERE     NOT FOR SOME f IN Familiari
              WHERE f.CapoFamiglia = i.Codice:
                    f.Sesso <> i.Sesso;
```

che in SQL diventa:

```
SELECT    i.NomeCognome
FROM      Impiegati i
WHERE     NOT EXISTS(SELECT *
                     FROM   Familiari f
                     WHERE  f.CapoFamiglia = i.Codice
                            AND f.Sesso <> i.Sesso);
```

La soluzione trova anche gli impiegati senza familiari a carico. Per escluderli si aggiunge una giunzione:

```
SELECT    DISTINCT i.NomeCognome
FROM      Impiegati i, Familiari f
WHERE     i.Codice = f.CapoFamiglia
AND       NOT EXISTS (SELECT *
                      FROM    Familiari
                      WHERE   CapoFamiglia = i.Codice
                              AND Sesso <> i.Sesso );
```

o) La soluzione con il quantificatore universale è:

```
SELECT    i.NomeCognome
FROM      Impiegati i
WHERE     FOR ALL x IN Partecipazioni WHERE x.Impiegato = 300:
              (FOR SOME y IN Partecipazioni
                   WHERE y.Impiegato = i.Codice:
                         x.Progetto = y.Progetto);
```

con la doppia negazione si ottiene:

```
SELECT    i.NomeCognome
FROM      Impiegati i
WHERE     NOT FOR SOME x IN Partecipazioni
                   WHERE x.Impiegato = 300:
              (NOT FOR SOME y IN Partecipazioni
                   WHERE y.Impiegato = i.Codice:
                         x.Progetto = y.Progetto);
```

che in SQL diventa:

```
SELECT    i.NomeCognome
FROM      Impiegati i
WHERE     NOT EXISTS
              (SELECT *
               FROM    Partecipazioni x
               WHERE   x.Impiegato =300
                       AND NOT EXISTS
                           (SELECT *
                            FROM    Partecipazioni y
                            WHERE   y.Impiegato = i.Codice
                                    AND x.Progetto = y.Progetto));
```

p)
```
CREATE    VIEW DipConAlmenoDueImp AS
SELECT    Numero, Nome
FROM      Impiegati, Dipartimenti
WHERE     Afferisce = Numero
GROUP BY  Numero, Nome
HAVING    COUNT(*) > 2;

SELECT    Nome, COUNT(*) AS NImp
FROM      Impiegati, DipConAlmenoDueImp
WHERE     Afferisce = Numero AND AnnoNascita > 1950
GROUP BY  Numero, Nome;
```

Il risultato non contiene i dipartimenti con più di due impiegati nati prima del 1950, che si trovano con l'interrogazione:

```
SELECT    *
FROM      DipConAlmenoDueImp
WHERE     FOR ALL x IN Impiegati WHERE x.Afferisce = Numero:
              x.AnnoNascita <= 1950;
```

con la doppia negazione si ottiene:

```
SELECT    *
FROM      DipConAlmenoDueImp
WHERE     NOT FOR SOME x IN Impiegati
              WHERE x.Afferisce = Numero:
                  x.AnnoNascita > 1950;
```

che in SQL diventa:

```
SELECT    Nome, 0 AS NImp
FROM      DipConAlmenoDueImp
WHERE     NOT EXISTS(SELECT *
                     FROM   Impiegati x
                     WHERE  x.Afferisce = Numero
                     AND    x.AnnoNascita > 1950);
```

Quindi la soluzione completa è la seguente:

```
SELECT      Nome, COUNT(*) AS NImp
FROM        Impiegati, DipConAlmenoDueImp
WHERE       Afferisce = Numero AND AnnoNascita > 1950
GROUP BY    Numero, Nome
  UNION
SELECT      Nome, 0 AS NImp
FROM        DipConAlmenoDueImp
WHERE       NOT EXISTS(SELECT *
                       FROM   Impiegati x
                       WHERE  x.Afferisce = Numero
                       AND    x.AnnoNascita > 1950);
```

Capitolo 7

SQL PER DEFINIRE E AMMINISTRARE BASI DI DATI

Esercizio 7.1 Si definisca uno schema relazionale con i comandi **CREATE TABLE** per trattare le informazioni e i vincoli d'integrità sui dipendenti di un'azienda, con attributi CodiceFiscale, Nome, AnnoAssunzione e Salario, e sui loro familiari a carico, con attributi Nome, AnnoNascita e RelazioneDiParentela.

Soluzione 7.1 Da Fare.

Esercizio 7.2 Si supponga che sia stato definito uno schema relazionale con le istruzioni:

CREATE TABLE R (K **CHAR**(8) **NOT NULL**, A **CHAR**(8), B **CHAR**(8))
PRIMARY KEY(K)

GRANT SELECT ON R **TO** caio

e siano stati immessi dei dati in R. Usando i comandi:

DROP TABLE Nome;
CREATE TABLE Nome (Attributo Tipo, ...) **AS** ExprSQL'
CREATE VIEW Nome (Attributo, ...) **AS** ExprSQL;
GRANT SELECT ON Nome **TO** Utente;

mostrare come si possa modificare lo schema in modo che i dati di R vengano memorizzati esclusivamente nelle tabelle:

R1(K **CHAR**(8) **NOT NULL**, A **CHAR**(8))
R2(K **CHAR**(8) **NOT NULL**, B **CHAR**(8))

e l'utente "caio" possa continuare a lavorare sulla base di dati come se esistesse la tabella:

R(K **INTEGER**(8) **NOT NULL**, A **INTEGER**(8), B **INTEGER**(8)).

Soluzione 7.2 Da Fare.

Esercizio 7.3 Definire lo schema per la base di dati relazionale ottenuta dall'Esercizio 5.3.

Soluzione 7.3 Da Fare.

Esercizio 7.4 Si mostri come trattare il vincolo di chiave esterna con i *trigger*.

Soluzione 7.4 Da Fare.

Esercizio 7.5 Si ricorda che date due sottoclassi C_1 e C_2 di una classe C, diciamo che:

a) C_1 e C_2 soddisfano il vincolo di copertura di C se $C_1 \cup C_2 = C$;
b) C_1 e C_2 soddisfano il vincolo di disgiunzione se non hanno nessun elemento in comune.

In generale si possono quindi avere quattro tipi di situazioni:

a) copertura disgiunta o partizione (vincolo di copertura e di disgiunzione);
b) copertura non disgiunta (solo vincolo di copertura);
c) sottoinsiemi disgiunti (solo vincolo di disgiunzione);
d) sottoinsiemi non disgiunti (nessun vincolo).

Si supponga di rappresentare C_1, C_2 e C con tre relazioni RC_1, RC_2 ed RC, con RC_1 ed RC_2 che contengono gli attributi propri di C_1 e C_2 e una chiave esterna per RC. Si supponga di poter dichiarare nello schema solo il vincolo di chiave primaria e di chiave esterna. Si mostri se è possibile rappresentare i vincoli dei quattro tipi di sottoclassi con il comando **CREATE TABLE**.

Soluzione 7.5 Da Fare.

Esercizio 7.6 Si risolva l'esercizio precedente usando i *trigger*.

Soluzione 7.6 Da Fare.

Capitolo 8

SQL PER PROGRAMMARE LE APPLICAZIONI

Esercizio 8.1 Si consideri il seguente frammento di codice SQLJ:

```
#sql [contesto]
        SELECT   Ammontare INTO  :ammontare
        FROM     Ordini
        WHERE    CodiceAgente = :numAgente
```

L'elaborazione del codice procede in tre fasi: (a) precompilazione dei frammenti SQL in Java, (b) compilazione del programma Java risultante, (c) esecuzione. Durante l'esecuzione, il controllo si alterna tra macchina astratta Java e il DBMS. Esemplifichiamo ora alcuni motivi per cui tale codice potrebbe essere scorretto. Immaginando che ogni errore sia scoperto prima possibile, specificare chi dei quattro attori (precompilatore, compilatore, macchina astratta Java e DBMS) segnala ciascun errore.

a) Sintassi SQL: il programmatore potrebbe scrivere WEHRE anziché **WHERE**.
b) Nomi: il programmatore potrebbe avere sbagliato a scrivere il nome della relazione, oppure quello dell'attributo Ammontare, oppure quello della variabile :ammontare.
c) Tipi: il tipo di Ammontare e quello di :ammontare potrebbero essere incompatibili.
d) Variazione dello schema: quando il programma viene eseguito la relazione Ordini potrebbe essere stata cancellata, oppure il tipo dell'attributo Ammontare potrebbe essere cambiato.
e) Univocità: il valore di :NumAgente potrebbe non essere associato ad alcun agente, oppure essere associato a più agenti.

Soluzione 8.1 Da Fare.

Esercizio 8.2 Si consideri la versione API del frammento di codice dell'esempio precedente.

```
PreparedStatement pstmt =
    con.prepareStatement(
```

```
        "SELECT Ammontare INTO :ammontare
          FROM Ordini WHERE CodiceAgente = ?");
pstmt.setString(1, codAgente);
risultato = pstmt.executeQuery();
```

In questo caso l'elaborazione di tale frammento procede come segue: compilazione del programma Java, esecuzione da parte della macchina astratta Java che interagisce con il DBMS. Anche in questo caso si cerchi di individuare in quale momento verrebbe scoperto ciascuno degli errori sopra elencati.

Soluzione 8.2 Da Fare.

Esercizio 8.3 Specificare vantaggi e svantaggi della programmazione di applicazioni usando un linguaggio integrato anziché un'API.

Soluzione 8.3 Da Fare.

Esercizio 8.4 Si considerino le seguenti applicazioni in ambito bancario. Indicare il livello di isolamento più opportuno per ciascuna di esse, spiegando la risposta.

a) Per ogni cliente della banca, contare le operazioni effettuate negli ultimi 500 giorni, ed aggiungere il nome del cliente ad un elenco se le operazioni sono più di trecento. L'elenco servirà a scopi di marketing.
b) Effettuare un trasferimento fondi, sottraendo un ammontare da un conto per aggiungerlo ad un altro.
c) Gestire un prelievo allo sportello come segue: il cassiere legge sul terminale il saldo corrente del cliente; se questo supera la cifra richiesta dal cliente, il cassiere comunica al sistema la cifra ed effettua il pagamento (specificare quali di queste operazioni sarebbero racchiuse nella transazione).

Soluzione 8.4 Da Fare.

Capitolo 9

REALIZZAZIONE DEI DBMS

Esercizio 9.1 Dire quali delle seguenti affermazioni è vera o falsa e giustificate la risposta:

a) I seguenti piani di accesso per l'interrogazione
SELECT A **FROM** R **ORDER BY** A
non sono equivalenti:

b) Operatore logico e operatore fisico sono sinonimi.
c) Ad ogni interrogazione SQL corrispondono più alberi logici.
d) Ad ogni albero logico corrispondono più alberi fisici.
e) La gestione della concorrenza con il blocco dei dati non crea condizioni di stallo.
f) Con il metodo disfare–rifare, nessun dato modificato da una T può essere riportato nella BD prima che il corrispondente record del giornale sia scritto nella memoria permanente.
g) Con il metodo rifare, tutte le modifiche di una T devono essere riportate nella BD prima che il record di commit sia scritto nel giornale.

Soluzione 9.1

a) *Falso*. Il primo piano esegue l'interrogazione con un algoritmo che recupera prima i record di R, poi li proietta su A e infine li ordina. Il secondo piano usa un algoritmo che prima ordina R poi recupera i record ordinati e infine li proietta su A, producendo lo stesso risultato.
b) *Falso*. Un operatore logico è un operatore dell'algebra relazionale, mentre un operatore fisico è un algoritmo per realizzare un operatore logico sfruttando le strutture di memorizzazione disponibili nella macchina fisica.

c) *Vero*. L'albero logico iniziale può essere trasformato in modi diversi applicando le regole di equivalenza dell'algebra relazionale.
d) *Vero*. Un operatore logico può essere realizzato con algoritmi diversi.
e) *Falso*. La tecnica del blocco dei dati può creare situazioni di stallo.
f) *Vero*. Il metodo disfare–rifare segue le regole per *disfare* e per *rifare*: prima di modificare la BD scrive nel giornale la vecchia versione e la nuova versione dei dati modificati.
g) *Falso*. Il metodo rifare non prevede che le modifiche delle transazioni siano riportate nella BD prima che il record di commit sia scritto nel giornale.

Esercizio 9.2 Si considerino due relazioni unarie $R(A)$ e $S(B)$ con i seguenti record:
$R = \{(A := 7), (A := 2), (A := 8), (A := 3), (A := 1), (A := 3), (A := 6)\}$
$S = \{(B := 4), (B := 2), (B := 1), (B := 3), (B := 2), (B := 7), (B := 3)\}$
Mostrare (a) il contenuto di un indice sull'attributo A di R e (b) il risultato prodotto dalla giunzione $R \underset{A=B}{\bowtie} S$ usando il NestedLoop.

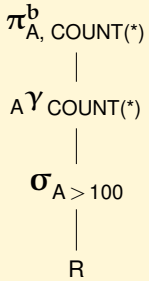
Soluzione 9.2 Da fare.

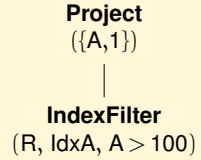
Esercizio 9.3 Si consideri la relazione R(A, B, C), con chiave primaria A, e l'interrogazione:

```
SELECT      A, COUNT(*)
FROM        R
WHERE       A > 100
GROUP BY    A;
```

Dare l'albero logico iniziale dell'interrogazione e un possibile piano di accesso.
Come cambia la soluzione se nella **SELECT** ci fosse **DISTINCT** A, **COUNT**$(*)$?

Soluzione 9.3



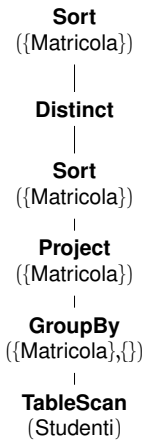


Non occorre raggruppare perché A è chiave. La soluzione non cambia se nella **SELECT** ci fosse **DISTINCT**.

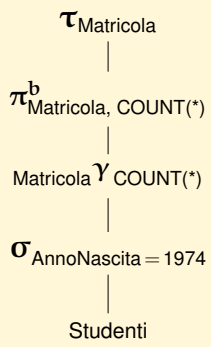
Esercizio 9.4 Si consideri la relazione Studenti(Matricola, Nome, AnnoNascita), ordinata sulla chiave primaria Matricola, e l'interrogazione:

```
SELECT      DISTINCT Matricola, COUNT(*)
FROM        Studenti
WHERE       AnnoNascita = 1974
GROUP BY    Matricola
ORDER BY    Matricola;
```

Dare l'albero logico iniziale dell'interrogazione e si dica se il seguente piano d'accesso produce il risultato cercato. Se non va bene, lo si modifichi in tre modi:
a) aggiungendo prima solo le parti mancanti (operatori e parametri),
b) semplificando poi il piano eliminando operatori inutili e
c) modificando infine il piano supponendo che esita un indice su AnnoNascita:

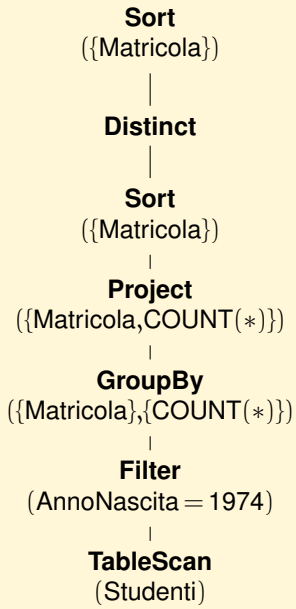


Soluzione 9.4
Albero logico iniziale dell'interrogazione:



Il piano di accesso non produce il risultato dell'interrogazione.

a) Completiamo il piano di accesso aggiungendo le parti mancanti, che sono il filtro su AnnoNascita e la funzione di aggregazione COUNT(*):



b) Adesso eliminiamo gli operatori superflui:

Project
({Matricola,1})
|
Filter
(AnnoNascita = 1974)
|
TableScan
(Studenti)

Partendo dal basso si eliminano:

- **GroupBy**: l'attributo di raggruppamento è chiave e quindi non serve raggruppare.
- **Project**:: l'operatore **GroupBy** restituisce record con campi Matricola, COUNT(*) e quindi il **Project** è inutile,
- **Sort**: la tabella è memorizzata ordinata sulla chiave Matricola e quindi non occorre ordinare,
- **Distinct**: tra gli attributi dei record che arrivano all'operatore **Distinct** è compresa una chiave, dunque non possono esistere record uguali,
- **Sort**: non serviva prima, non serve adesso.

c) Considerando l'indice su AnnoNascita il piano diventa:

Sort
({Matricola})
|
Project
({Matricola,1})
|
IndexFilter
(Studenti, IdxAN, AnnoNascita = 1974)

Esercizio 9.5 Si consideri il seguente schema relazionale:

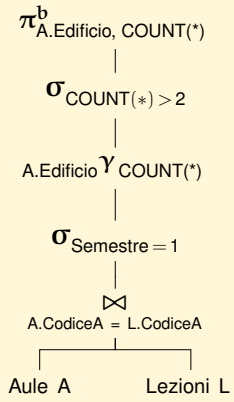
Aule(CodiceA, Edificio, Capienza)
Lezioni(CodiceA, CodiceC, Ora, GiornoSett, Semestre)
Corsi(CodiceC, NomeC, Docente)

e l'interrogazione:

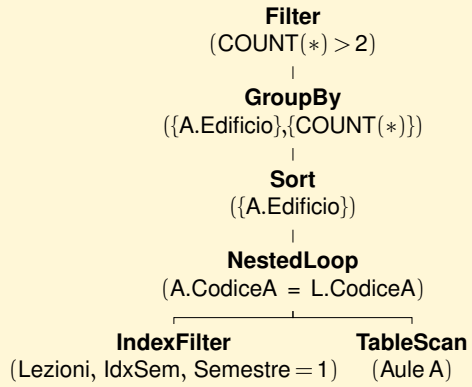
```
SELECT      A.Edificio, COUNT(*)
FROM        Aule A, Lezioni L
WHERE       A.CodiceA = L.CodiceA AND Semestre = 1
GROUP BY    A.Edificio
HAVING      COUNT(*) > 2;
```

Si dia l'albero logico iniziale dell'interrogazione e un piano di accesso che utilizzi indici:
a) nel caso di giunzione con l'operatore NestedLoop e
b) nel caso di giunzione con l'operatore IndexNestedLoop.

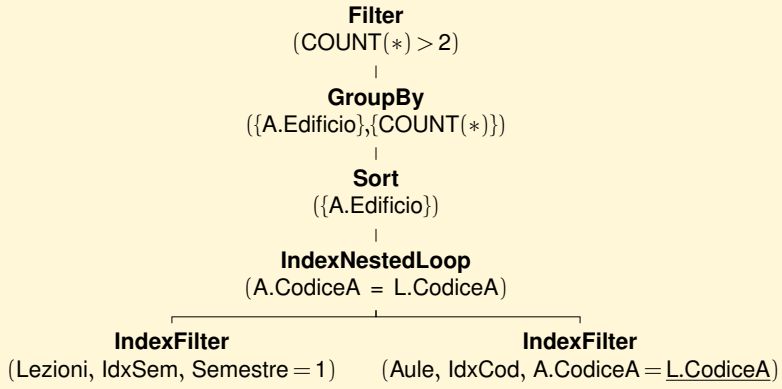
Soluzione 9.5
Albero logico iniziale dell'interrogazione:



a) *Piano di accesso con* **NestedLoop**:



b) *Piano di accesso con* **IndexNestedLoop**:



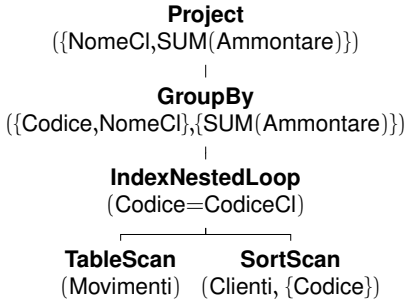
Esercizio 9.6 Si consideri la base di dati:

Clienti(Codice, NomeCl, AnnoNascita), con chiave primaria Codice
Movimenti(CodiceCl, Ammontare, Tipo), con chiave esterna CodiceCl

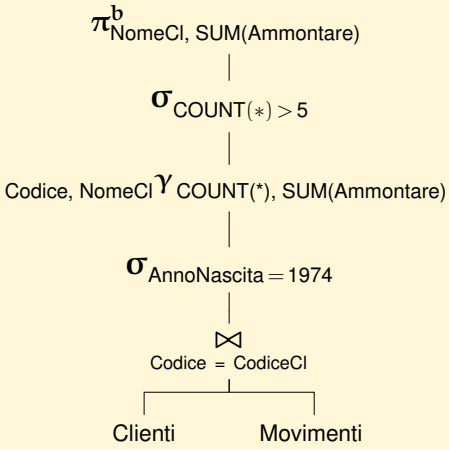
e l'interrogazione:

SELECT NomeCl, **SUM**(Ammontare)
FROM Clienti, Movimenti
WHERE Codice = CodiceCl **AND** AnnoNascita = 1974
GROUP BY Codice, NomeCl
HAVING **COUNT**(∗) > 5 ;

Dare l'albero logico iniziale dell'interrogazione e si dica
a) se il seguente piano d'accesso è corretto,
b) se produce il risultato cercato.
Se non va bene, lo si modifichi aggiungendo le parti mancanti (operatori e parametri).



Soluzione 9.6
Albero logico iniziale dell'interrogazione:



a) Il piano non è corretto perché mancano i filtri per AnnoNascita = 1974 e per **HAVING**, il **GROUP BY** richiede i record ordinati su Codice, NomeCl, il **SortScan** deve essere un **IndexFilter** per la presenza dell'**IndexNestedLoop**, e quindi b) non produce il risultato cercato. Un piano corretto è: