Basi di Dati I, 25 gennaio 2021 - Seconda prova intercorso

Adriano Peron

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: adrperon@unina.it

Si consideri il seguente schema relazionale che descrive la collocazione spaziale di località assumendo un sistema di riferimento cartesiano.

$LOCALITA(\underline{CodL}, Nome, Tipo, CodNode)$
$NODO(\underline{CodNodo}, AngX, AngY)$
$PERCORSO(\underline{CodPer}, Tipo, NodoI, NodoF)$
$INPERCORSO(CodPer, Ordine, Nodo)$

$LOCALITA$ descrive le località di interesse: il tipo indica la tipologia della località (città, comune, frazione etc.); CodNode fornisce il nodo geografico associato alla località. $NODO$ fornisce le coordinate AngX ed AngY per il riferimento spaziale dei nodi geografici. $PERCORSO$ descrive percorsi tra un nodo iniziale (NodoI) e un nodo finale (NodoF) indicando il tipo di percorso (ad es. strada statale, autostrada, lineaferroviaria, etc) I nodi che fanno parte di un percorso sono elencati in $INPERCORSO$ dove si indica quali nodi sono associati ad un percorso e la posizione del nodo nel percorso (0 nodo iniziale, 1 primo nodo del percorso etc.)

Esercizio 01 *(30 minuti) Si supponga di avere una tabella*
$Distanze(codloc1, codloc2, idpercorso, distanza)$
che contiene le distanze tra due località codloc1 e codloc2 che si trovano su uno stesso percorso avente idpercorso. Si scriva una procedura che riceve come parametro di ingresso il codice di una località. La procedura verifica quali siano le località raggiungibili da quella fornita in uno stesso percorso. Se non esiste già una riga nella tabella Distanze per la coppia di locazioni nel percorso, la procedura calcola la distanza delle due località e inserisce una riga nella tabella Distanze. La distanza tra due localita consecutive in un percorso viene fornita invocando una funzione $dist(point1, point2)$ *che si suppone data e che prende in ingresso due codici di punti e restituisce un numerico corrispondente. La distanza tra due località non consecutive in un percorso è data dalla somma delle distanze tra le coppie consecutive di località nel percorso che portano dalla prima localit di interesse all'ultima localit di interesse.*

Esercizio 02 *(30 minuti) Si scriva una funzione in SQL DINAMICO che riceve in ingresso una stringa di codici di percorsi separati dal carattere +. La funzione restituisce la stringa di codici dei nodi che sono inclusi in TUTTI i percorsi forniti nella stringa parametro.*

Basi di Dati I, 25 gennaio 2021 - II Prova Intercorso

Adriano Peron

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: adrperon@unina.it

Si consideri il seguente schema relazionale che descrive la collocazione spaziale di località assumendo un sistema di riferimento cartesiano.

$LOCALITA(\underline{CodL}, Nome, Tipo, CodNode)$
$NODO(\underline{CodNodo}, AngX, AngY)$
$PERCORSO(\underline{CodPer}, Tipo, NodoI, NodoF)$
$INPERCORSO(CodPer, Ordine, Nodo)$
$DISTANZE(codloc1, codloc2, idpercorso, distanza)$

$LOCALITA$ descrive le località di interesse: il tipo indica la tipologia della località (città, comune, frazione etc.); CodNode fornisce il nodo geografico associato alla località. $NODO$ fornisce le coordinate AngX ed AngY per il riferimento spaziale dei nodi geografici. $PERCORSO$ descrive percorsi tra un nodo iniziale (NodoI) e un nodo finale (NodoF) indicando il tipo di percorso (ad es. strada statale, autostrada, lineaferroviaria, etc) I nodi che fanno parte di un percorso sono elencati in $INPERCORSO$ dove si indica quali nodi sono associati ad un percorso e la posizione del nodo nel percorso (0 nodo iniziale, 1 primo nodo del percorso etc.). $DISTANZE$ contiene le distanze tra due località codloc1 e codloc2 che si trovano su uno stesso percorso avente idpercorso.

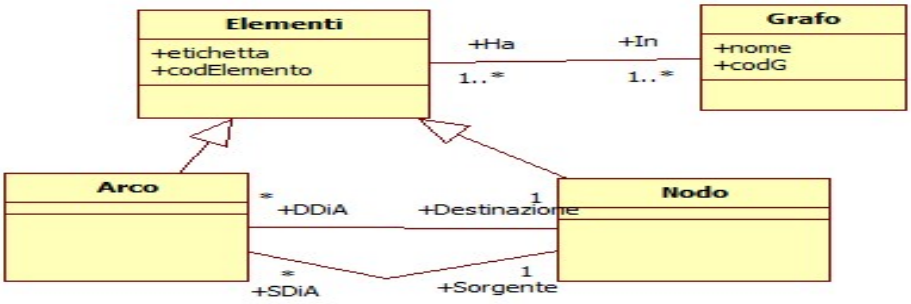
Esercizio 01 *Si supponga di avere una tabella*
$CONNESSIONE(codloc1, idpercorso1, codloc2, idpercorso2, distanza)$
inizialmente vuota che dovrebbe contenere la distanza tra due locazioni appartenenti a su percorsi diversi (codloc1 nel percorso idpercorso1 e codloc2 nel percorso idpercorso2). Si scriva una procedura ha come effetto il popolamento della tabella $CONNESSIONE$. Devono essere inserite le distanze tra tutte le coppie di locazioni (A,B) appartenenti a diversi percorsi che hanno una locazione in comune C (da A è possibile raggiungere C mediante un cambio di percorso nella locazione C). La distanza si ottiene sommando le distanze delle tratte disponibili nella tabella $DISTANZE$ (A-C + C-B).

Esercizio 02 *(Punti 7, 25 minuti) Si scriva una funzione in SQL DINAMICO che riceve in ingresso una stringa di codici di locazioni separati dal carattere +. La funzione restituisce la stringa di codici di percorso dei percorsi in cui sono incluse TUTTE le locazioni nella stringa parametro.*

Basi di Dati I, Esempio prova intercorso

Adriano Peron

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: adrperon@unina.it



Esercizio 01 *(Punti 9, 30 minuti) Si consideri la bozza di Class Diagram in figura che descrive grafi composti di nodi e di archi (un arco ha un nodo sorgente e uno destinazione). CodElem e CodG sono attibuti chiave. La specializzazione è totale e disgiunta. Si scriva prima lo schema logico per il class diagram e si definiscano poi le tabelle.*

Si consideri il seguente schema relazionale che descrive un frammento della base di dati per gestire gli accessi ad una biblioteca digitale di periodici. Le riviste sono identificate dal codice $ISNN$ e sono strutturate in fascicoli (identificati da $CodF$). Ogni fascicolo contiene articoli identificati dall'attibuto Doi. L'utente identificato dal codice fiscale CF ha un profilo associato $Codprofilo$ che regola le possibilità di accesso (numero di articoli consultati al giorno e al mese). In $ACCESSO$ ogni istanza memorizza l'accesso di un utente an un articolo. In $PAROLECHIAVE$ viene memorizzato un insieme di parole chiave significative per una rivista. Le parole chiave sono usate per descrivere astrattamente gli articoli in base al loro contenuto. L'associazione tra parole chiave e articoli viene memorizzata nella relazione $DESCRIZIONE$.

$RIVISTA(\underline{ISNN}, Titolo, Editore, Perodicita)$
$FASCICOLO(\underline{CodF}, Titolo, ISNN, Anno, Numero)$
$ARTICOLO(\underline{Doi}, CodF, Titolo, Autore, Sommario, PagI, PagF)$
$UTENTE(\underline{CF}, email, CodProfilo, Nome, Cognome, DataN)$
$ACCESSO(CF, Doi, Data)$
$PROFILO(\underline{CodProfilo}, Tipo, MaxGiorno, MaxMese)$
$PAROLECHIAVE(Parola, ISNN)$
$DESCRIZIONE(Parola, Doi)$

Esercizio 02 *(Punti 8, 20 minuti) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce l'ISNN della rivista che ha avuto nel 2019 il maggior numero di accessi.*

Esercizio 03 *(Punti 8, 20 minuti) Si scriva una interrogazione in SQL che restituisca coppie di utenti che nel 2019 hanno fatto lo stesso numero di accessi.*

Esercizio 04 *(Punti 8, 20 minuti) Si implementino nel modo più adeguato i seguenti vincoli:*

1. *Un utente non può avere più accessi in un anno in corso di quanto previsto dal suo profilo.*
2. *Se è presente in $DESCRIZIONE$ una parola chiave associata ad un articolo, allora la parola chiave deve essere associata alla rivista che contiene l'articolo.*
3. *Un utente può avere un'unica email*

Basi di Dati I, 16 luglio 2021

Adriano Peron

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: adrperon@unina.it

Si consideri il seguente schema relazionale che descrive i dati di gestione in un sistema per l'accesso parralelo. L'intuizione è che una transazione (sequenza atomica di operazioni) prima di poter operare su una risorsa deve chiedere l'autorizzazione all'accesso (in lettura o scrittura) e solo quando la risorsa le viene assegnata può operarae per leggere o modificare il valore della risorsa. Le operazioni possono essere $CREA$ (crea una nuova risorsa), CANCELLA (cancella una risorsa) MODIFICA (modifica il valore della risorsa) COMMIT (chiusura della transazione) ABORT (annulla una transazione). Le risorse condivise son identificate dall'attributo $CodRisorsa$ e lo stato indica se sono libere $UNLOCK$, in uso in lettura $R-LOCK$ o in uso in scrittura $W-LOCK$. Le transazioni identificate da $CodTransazione$ inoltrano richieste di operazioni che sono registrate nella tabella $RICHIESTE$. Ogni richiesta ha un tempo di inoltro, un tipo di accesso (o R-LOCK o W-LOCK) e la risorsa richiesta. Nella tabella ASSEGNAZIONE sono memorizzate le assegnazioni correnti delle risorse alle transazioni. Nella tabella LOG vengono invece registrate le operazioni svolte dalle transazioni sulle risorse a loro assegnate. In particolare, per ogni operazione $MODIFICA$ si registra il valore della risorsa prima e dopo l'operazione; per operazione CANCELLA si esprime solo $ValorePrima$; per CREA si esprime solo il valore $ValoreDopo$ e per COMMIT ed ABORT e CHECK non si esprime nessun valore (NULL). In aggiunta per l'operazione CHECK non si esprime neppure il codice della transazione (il record di CHECK rappresenta un punto di controllo del sistema).

$LOG(\underline{Cod}, Operazione, CodRisorsa, ValorePrima, ValoreDopo, CodTransazione)$
$RISORSA(\underline{CodRisorsa}, Locazione, Valore, Stato)$
$RICHIESTE(CodTransazione, Tempo, tipoAccesso, CodRisorsa)$
$ASSEGNAZIONE(CodTransazione, Tempo, CodRidorsa, tipoAccesso)$

Esercizio 01 *(Punti 8) Si scriva una interrogazione in algebra relazionale che se valutata fornisce le transazioni che non hanno registrato la operazione di COMMIT sul log (sono ancora attive) e non hanno risorse assegnate.*

Esercizio 02 *(Punti 8) Scrivere una interrogazione (vista) che considera le operazioni successive all'ultimo $CHECK$ (solo operazioni successive per tempo al tempo dell'operazione di CHECK più recente nel LOG) e per quelle operazioni produce il seguente conteggio* $(tempo, N\_transazioni, N\_Commit, N\_Abort, N\_risorse\_riscritte)$ *dove tempo è il valore dell'ultimo CHECK, $N\_transazioni$ il numero di transazioni distinte dopo l'ultimo $CHECK$, N_Commi e N_Abort il numero di operazioni di COMMIT e ABORT, $N\_risorse\_riscritte$ il numero di risorse distinte riscritte.*

Esercizio 03 *(Punti 8) Si implementi il seguente trigger. Quando una transazione registra una operazione di ABORT sul log, tutte le scritture fatte dalla transazione e riportate sul LOG devono essere annullate in ordine inverso a quelle in cui sono state fatte. Per annullare le scritture si deve consultare il log e si deve assegnare ad ogni risorsa scritta dalla transizione il valore ValorePrima riportato nel LOG. Inoltre, le risorse assegnate alla transazione devono tornare libere: si rimuovono le assegnazioni alla transazione e lo stato della risorsa assume valore UNLOCK.*

Esercizio 04 *(Punti 8) Si scriva una funzione con parametro intero Tout che per tutte le transazioni T1 che sono in attesa per una risorsa per un tempo superiore a Tout (differenza tra il tempo di registrazione della richiesta e il tempo corrente) controlli se ci sia un deadlock (cioè se esiste un'altra transazione T2 che occupa la risorsa richiesta e la transazione T2 richiede una risorsa asegnata alla transazione T1). La funzione restituisce una stringa coi codici delle transazioni T1 in deadlock cosi trovate.*

Basi di Dati I, 16 luglio 2021

Adriano Peron

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: adrperon@unina.it

Si consideri il seguente schema relazionale che descrive i dati di gestione in un sistema per l'accesso parralelo. L'intuizione è che una transazione (sequenza atomica di operazioni) prima di poter operare su una risorsa deve chiedere l'autorizzazione all'accesso (in lettura o scrittura) e solo quando la risorsa le viene assegnata può operarae per leggere o modificare il valore della risorsa. Le operazioni possono essere $CREA$ (crea una nuova risorsa), CANCELLA (cancella una risorsa) MODIFICA (modifica il valore della risorsa) COMMIT (chiusura della transazione) ABORT (annulla una transazione). Le risorse condivise son identificate dall'attributo $CodRisorsa$ e lo stato indica se sono libere $UNLOCK$, in uso in lettura $R-LOCK$ o in uso in scrittura $W-LOCK$. Le transazioni identificate da $CodTransazione$ inoltrano richieste di operazioni che sono registrate nella tabella $RICHIESTE$. Ogni richiesta ha un tempo di inoltro, un tipo di accesso (o R-LOCK o W-LOCK) e la risorsa richiesta. Nella tabella ASSEGNAZIONE sono memorizzate le assegnazioni correnti delle risorse alle transazioni. Nella tabella LOG vengono invece registrate le operazioni svolte dalle transazioni sulle risorse a loro assegnate. In particolare, per ogni operazione $MODIFICA$ si registra il valore della risorsa prima e dopo l'operazione; per operazione CANCELLA si esprime solo $ValorePrima$; per CREA si esprime solo il valore $ValoreDopo$ e per COMMIT ed ABORT e CHECK non si esprime nessun valore (NULL). In aggiunta per l'operazione CHECK non si esprime neppure il codice della transazione (il record di CHECK rappresenta un punto di controllo del sistema).

$LOG(\underline{Cod}, Operazione, CodRisorsa, ValorePrima, ValoreDopo, CodTransazione)$
$RISORSA(\underline{CodRisorsa}, Locazione, Valore, Stato)$
$RICHIESTE(CodTransazione, Tempo, tipoAccesso, CodRisorsa)$
$ASSEGNAZIONE(CodTransazione, Tempo, CodRidorsa, tipoAccesso)$

Esercizio 01 *Si scriva una interrogazione in algebra relazionale che se valutata fornisce le transazioni che non hanno registrato la operazione di COMMIT sul log (sono ancora attive) e non hanno risorse assegnate.*

Esercizio 02 *Si scriva la stessa interrogazione precedente in SQL.*

Esercizio 03 *Scrivere una interrogazione (vista) che considera le operazioni successive all'ultimo $CHECK$ (solo operazioni successive per tempo al tempo dell'operazione di CHECK più recente nel LOG) e per quelle operazioni produce il seguente conteggio*
$(tempo, N\_transazioni, N\_Commit, N\_Abort, N\_risorse\_riscritte)$ dove tempo è il valore dell'ultimo CHECK, $N\_transazioni$ il numero di transazioni distinte dopo l'ultimo $CHECK$, N_Commi e N_Abort il numero di operazioni di COMMIT e ABORT, $N\_risorse\_riscritte$ il numero di risorse distinte riscritte.

Esercizio 04 *Si scriva nel modo più opportuno il seguente insieme di vicoli indicando la tipologia di vincolo:*

1. *Se una risorsa viene assegnata in scrittura a una transazione non può essere assegnata a nessuna altra transazione.*
2. *Non ci possono essere richieste di una transazione sulla stessa risorsa con tipi di accesso diversi.*
3. *Se una transazione ha registrato un COMMIT sul log non può avere più richieste o assegnazioni dopo la registrazione del COMMIT.*

Basi di Dati I, Esercizi

Adriano Peron

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: adrperon@unina.it

Si consideri il seguente schema relazionale che descrive la gestione dei piani di studio degli studenti:
$STUDENTE(\underline{Matricola}, Nome, Cognome)$
$PIANI(\underline{CodPiano}, Matricola, Data)$
$COMPOSIZIONE(CodPiano, CodI, Anno, Voto, Lode)$
$INSEGNAM(\underline{CodI}, Titolo, CFU, Tipo)$
$ANNI(CodEsame, Anno)$
$PROPED(CodEsame, CodEsameProP)$
$VERBALEESAME(\underline{CodV}, Data, CodI, CodDocente)$
$ESAMIVERBALE(CodV, Matricola, Voto, Lode)$

$PIANI$, fornisce la data di registrazione del Piano di Studi e la matricola dello studente (chiave esterna); $COMPOSIZIONE$ descrive la composizione di un piano di studio: una riga per ogni esame presente nel piano di studi insieme all'anno di corso in cui è previsto; Gli attributi *voto* e *Lode* sono parziali e viene assegnato loro un valore quando l'esame viene sostenuto. $INSEGNAM$ descrive gli attibuti degli insegnamenti; l'attributo tipo può assumere i valori $OBBLIGATORIO$, $FACOLTATIVO$, $DEFAULT$; $ANNI$ descrive i possibili anni del corso di studi in cui un insegnamento può essere collocato (in genere un insegnamento può essere collocato in diversi anni); $PROPED$ descrive i legami di propedeuticità tra gli insegnamenti. $VERBALEESAME$ contiene i verbali di esame per un insegnamento ($CodI$). Ogni verbale di esame contiene uno o più esami: gli esami presenti nel verbale sono descritti in $ESAMIVERBALE$.

Esercizio 01 *Si scriva una espressione in algebra relazionale che, se valutata, fornisca il nome e cognome e la matricola degli studenti che hanno un piano di studi dove TUTTI gli insegnamenti facoltativi presenti NON hanno richiesta di propedeuticità.*

Esercizio 02 *Definire nel modo più semplice il seguente insieme di vincoli:*

- *lo stesso insegnamento non deve comparire più di una volta in un piano di studi;*
- *nella tabella $PROPED$ ad un esame possono essere associati solo insegnamenti effettivamente presenti nella tabella $INSEGNAM$;*
- *la somma dei CFU degli insegnamenti presenti in ogni anno in un piano di studi deve essere esattamente 60 CFU.*
- *ogni piano di studi deve contenere tutti gli esami obbligtori.*

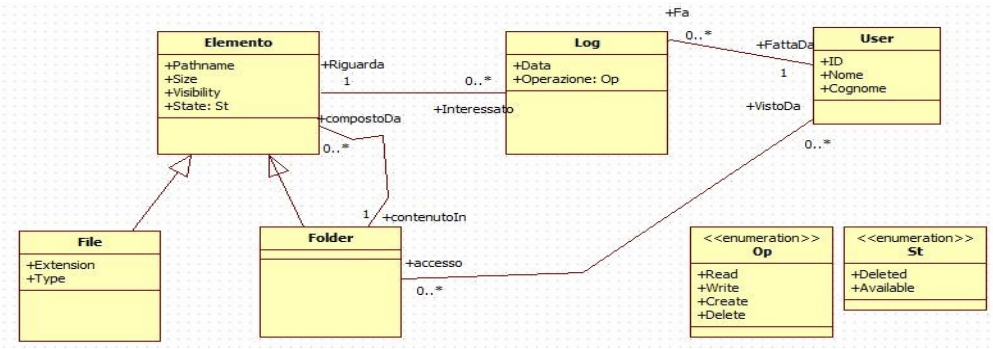
Esercizio 03 *Si scriva un trigger che viene attivato quando viene inserito un nuovo verbale di esame. L'effetto del trigger è il seguente. Per ogni esame presente nel verbale*

- *si registra il voto e la data nel piano di studi dello studente*

Esercizio 04 *Si scriva un trigger che viene attivato quando viene inserito un nuovo piano di studi per uno studente. L'effetto del trigger è il seguente. Per ogni esame presente nel verbale*

- *Si crea il piano di studi dello studente aggiungendo tutti gli esami obbligatori.*
- *Si aggiungono tutti gli esami di default collocati nel primo anno possibile.*

1 Progettazione



Esercizio 11 *Si consideri il class diagram riportato in figura che descrive un file system tradizionale con lementi del tipo File o Cartella con la usuale strutturazione ad albero delle cartelle. Un utente ha una relazione di visibilità sulle cartelle (le cartelle a cui può accedere). In una struttura di log sono memorizzate le operazioni fatte dagli utenti sugli elementi.*

1. *Si ristrutturi il class diagram per renderlo adeguato ad una traduzione nel modello dei dati relazionale indicando testualmente gli eventuali vincoli di ristrutturazione (6 punti);*
2. *Si forniscano gli schemi relazionali per il class diagram ristrutturato (7 punti);*
3. *Si definisca usando SQL le tabelle (5 punti);*
4. *Nella definizione delle tabelle si scrivano i vincoli: (a) Visibility è FALSE quando State è DELETED; (b) due elementi contenuti nello stesso folder hanno pathname diversi; (c) si dica il tipo dei due vincoli. (4 punti)*

Algebra relazionale. Basi di Dati I

Adriano Peron

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: adrperon@unina.it

Si consideri il seguente schema relazionale che descrive automi costituiti da stati e transizioni. Gli stati e le transizioni sono etichettati da stringhe di caratteri. Uno stato può essere iniziale o finale (gli attributi iniziale e finale sono parziali). La relazione $RAGGIUNGIBILE$ mantiene traccia della raggiungibilità tra stati di un automa. Una traccia è un cammino tra due stati (da Start a End) tale che concatenando le etichette delle transizioni separandole con un trattino si ottiene la stringa $Parola$. Ad esempio, se il cammino nell'automa A è $s_1 \xrightarrow{aa} s_2 \xrightarrow{bb} s_3 \xrightarrow{cc} s_4$ si ha in $RAGGIUNGIBILE$ una riga $(A, s_1, "aa-bb-cc", s_4)$. Una parola P si dice riconosciuta dall'automa A se esiste in $RAGGIUNGIBILE$ una riga (A, s, P, s') dove s è uno stato iniziale di A s' è uno stato finale di A.

$AUTOMA(\underline{CodA}, Descrizione)$
$STATO(\underline{CodS}, Etichetta, CodA, Iniziale, Finale)$
$TRANSIZIONE(\underline{CodT}, Etichetta, StatoIn, StatoOut, CodA)$
$RAGGIUNGIBILE(CodA, Start, Parola, End)$.

Esercizio 01 *Si scriva una interrogazione in algebra relazionale che se valutata fornisce i codici degli automi che non hanno self loop (transizioni cappio su un nodo).*

Esercizio 02 *Si scriva una interrogazione in algebra relazionale che se valutata fornisce i codici degli automi deterministici. Un automa è deterministico se non ha MAI due transizioni uscenti dallo stesso stato aventi la stessa etichetta.*

Esercizio 03 *Si scriva una interrogazione in algebra relazionale che restituisce i codici degli automi per cui per ogni coppia di stati (s1,s2) dell'automa esiste in RAGGIUNGIBILE una parola che porta da s1 a s2.*

Esercizio 04 *Si scriva in algebra relazionale una espressione che restituisce gli automi che riconoscono lo stesso insieme di parole dell'automa di codice XXX.*

Esercizio 05 *Si scriva in algebra relazionale una relazione che per ogni automa restituisca il numero di stati finali dell'automa, il numero di parole riconosciute presenti nella tabella RAGGIUNGIBILE, il numero di stati distinti raggiungibili da uno stato iniziale. (lo Schema da restituire ha la forma $RISULTATO(codA, Nstati, Nparole, Nragg)$).*

Esercizio 06 *Si scriva in algebra relazionale una espressione che restituisce l'automa con il massimo numero di stati.*

Basi di Dati e Sistemi Informativi I, 5 dicembre 2014 - Prova A

Facoltà di Scienze M.F.N., Corso di Laurea in Informatica

Si consideri il seguente schema relazionale che descrive un frammento della base di dati per gestire un applicazione tipo social network per la condivisione di Post (di tipo testuale).

$Post(\underline{IdPost}, Autore, Data, Ora, Testo)$
$Commento(\underline{IdPost, Ordine}, Testo, Autore, Data, Ora)$
$Key(parola, \underline{Tema})$
$Presenza(IdPost, Tema)$
$Sottoscrizione(IdUtente, Tema, Data)$
$Notifica(IdPost, IdUtente, Data, Ora)$
$Amici(IdUtente1, IdUtente2)$

Post descrive gli interventi nel social network; *Commento* indica i commenti ai Post, ordine indica l'ordine di inserimento del commento al post (Idpost e Ordine sono chiave). Key indica parole chiave rilevanti per individuare il tema del post ed il Tema generale a cui sono riconducibili (ad esempio, parola programmazione, tema informatica). Gli utenti possono esprimere il loro interesse per alcuni temi in modo da ricevere notifiche quando vengono pubblicati post correlati al tema (tabella *Sottoscrizione*). $Notifica$ indica le notifiche dei post agli utenti amici e interessati. *Amici* indica la relazione di amicizia tra coppie di utenti.

Esercizio 01 *(Punti 9 A) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce l'identificativo di post che sono stati commentati da tutti gli amici dell'autore.*

Esercizio 02 *(Punti 9 A) Si scriva una interrogazione in SQL che restituisca identificativo di post e tema del post tale che nel testo del post occorrono due parole diverse entrambe associate allo stesso tema.*

Esercizio 03 *(Punti 9 A) Si esprimano nel modo più opportuno i seguenti vincoli:*

1. *La relazione Amici è simmetrica (se (ut1,ut2) si trova nella relazione anche (ut2,ut1) si trova nella relazione);*
2. *Se un utente ha ricevuto una notifica o è amico dell'autore del post o ha sottoscritto le notifiche del tema del post;*
3. *Le notifiche di un post ad un utente vengono fatte una sola volta;*
4. *I commenti fanno riferimento a post presenti e non cancellati.*

Esercizio 04 *(Punti 9 A) Si scriva una vista che per ogni utente ed ogni mese fornisca un riepilogo dell'attività dell'utente del tipo*
(Utente,anno,mese,N_post,N_nocom, N_com_amici). (N_post il numero di post dell'utente nel mese, N_com_amici il numero di commenti di amici ai post del mese,N_nocom numero di post senza commenti nel mese)

Basi di Dati e Sistemi Informativi I, 5 dicembre 2014 - Prova B

Facoltà di Scienze M.F.N., Corso di Laurea in Informatica

Si consideri il seguente schema relazionale che descrive un frammento della base di dati per gestire un applicazione tipo social network per la condivisione di Post (di tipo testuale).

$Post(\underline{IdPost}, Autore, Data, Ora, Testo)$
$Commento(\underline{IdPost, Ordine}, Testo, Autore, Data, Ora)$
$Key(parola, Tema)$
$Presenza(IdPost, Tema)$
$Sottoscrizione(IdUtente, Tema, Data)$
$Notifica(IdPost, IdUtente, Data, Ora)$
$Amici(IdUtente1, IdUtente2)$

Post descrive gli interventi nel social network; *Commento* indica i commenti ai Post, ordine indica l'ordine di inserimento del commento al post (Idpost e Ordine sono chiave). Key indica parole chiave rilevanti per individuare il tema del post ed il Tema generale a cui sono riconducibili (ad esempio, parola programmazione, tema informatica). Gli utenti possono esprimere il loro interesse per alcuni temi in modo da ricevere notifiche quando vengono pubblicati post correlati al tema (tabella *Sottoscrizione*). $Notifica$ indica le notifiche dei post agli utenti amici e interessati. *Amici* indica la relazione di amicizia tra coppie di utenti.

Esercizio 01 *(Punti 9 B) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce l'identificativo di utenti autori di post che sono stati commentati solo da amici (mai da utenti non inclusi tra gli amici).*

Esercizio 02 *(Punti 9 B) Si scriva una interrogazione SQL che fornisce l'identificativo di post che sono stati commentati da tutti i sottoscrittori del tema del post.*

Esercizio 03 *(Punti 9 B) Si esprimano nel modo più opportuno i seguenti vincoli:*

1. *L'ordinamento dei commenti deve rispettare l'ordine di data e ora di pubblicazione (non può comparire prima nell'ordine ciò che è stato pubblicato dopo);*
2. *Se viene indicata la presenza di un tema in un post ci deve essere l'occorrenza nel testo di un post di una parola associata al tema;*
3. *Un utente può sottoscrivere solo una volta un tema;*
4. *I commenti fanno riferimento a post presenti e non cancellati.*

Esercizio 04 *(Punti 9 B) Si scriva una vista che per ogni utente ed ogni mese fornisca un riepilogo dell'attività dell'utente del tipo (Utente,anno,mese,N_post,N_nocom, N_com_nonamici). (N_post il numero di post dell'utente nel mese, N_com_nonamici il numero di commenti a post dell'utente nel mese fatti da utenti che non sono amici, N_nocom numero di post senza commenti)*

Basi di Dati e Sistemi Informativi I, 5 dicembre 2014 - Prova C

Facoltà di Scienze M.F.N., Corso di Laurea in Informatica

Si consideri il seguente schema relazionale che descrive un frammento della base di dati per gestire un applicazione tipo social network per la condivisione di Post (di tipo testuale).

$Post(\underline{IdPost}, Autore, Data, Ora, Testo)$
$Commento(\underline{IdPost, Ordine}, Testo, Autore, Data, Ora)$
$Key(parola, \underline{Tema})$
$Presenza(IdPost, Tema)$
$Sottoscrizione(IdUtente, Tema, Data)$
$Notifica(IdPost, IdUtente, Data, Ora)$
$Amici(IdUtente1, IdUtente2)$

Post descrive gli interventi nel social network; *Commento* indica i commenti ai Post, ordine indica l'ordine di inserimento del commento al post (Idpost e Ordine sono chiave). Key indica parole chiave rilevanti per individuare il tema del post ed il Tema generale a cui sono riconducibili (ad esempio, parola programmazione, tema informatica). Gli utenti possono esprimere il loro interesse per alcuni temi in modo da ricevere notifiche quando vengono pubblicati post correlati al tema (tabella *Sottoscrizione*). $Notifica$ indica le notifiche dei post agli utenti amici e interessati. *Amici* indica la relazione di amicizia tra coppie di utenti.

Esercizio 01 *(Punti 9 C) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce l'identificativo di post che sono stati commentati da tutti i sottoscrittori del tema del post.*

Esercizio 02 *(Punti 9 C) Si scriva una interrogazione in SQL che restituisca identificativo di post e due parole associate a temi distinti e il post sia tale che nel suo testo occorrono entrambe le parole.*

Esercizio 03 *(Punti 9 A) Si esprimano nel modo più opportuno i seguenti vincoli:*

1. *Se un utente ha ricevuto una notifica o è amico dell'autore del post o ha sottoscritto le notifiche del tema del post;*
2. *L'ordinamento dei commenti deve rispettare l'ordine di data e ora di pubblicazione (non può comparire prima nell'ordine ciò che è stato pubblicato dopo);*
3. *Le notifiche di un post ad un utente vengono fatte una sola volta;*
4. *Le notifiche fanno riferimento a post presenti e non cancellati.*

Esercizio 04 *(Punti 9 A) Si scriva una vista che per ogni utente ed ogni mese fornisca un riepilogo dell'attività dell'utente del tipo*
(Utente,anno,mese,N_post,N_nocom, N_com_sottoscrittori). (N_post il numero di post dell'utente nel mese, N_com_amici il numero di commenti di sottostrittori al tema del post,N_nocom numero di post senza commenti nel mese)

Basi di Dati e Sistemi Informativi I, 5 dicembre 2014 - Prova D

Facoltà di Scienze M.F.N., Corso di Laurea in Informatica

Si consideri il seguente schema relazionale che descrive un frammento della base di dati per gestire un applicazione tipo social network per la condivisione di Post (di tipo testuale).

$Post(\underline{IdPost}, Autore, Data, Ora, Testo)$
$Commento(\underline{IdPost, Ordine}, Testo, Autore, Data, Ora)$
$Key(parola, \overline{Tema})$
$Presenza(IdPost, Tema)$
$Sottoscrizione(IdUtente, Tema, Data)$
$Notifica(IdPost, IdUtente, Data, Ora)$
$Amici(IdUtente1, IdUtente2)$

Post descrive gli interventi nel social network; *Commento* indica i commenti ai Post, ordine indica l'ordine di inserimento del commento al post (Idpost e Ordine sono chiave). Key indica parole chiave rilevanti per individuare il tema del post ed il Tema generale a cui sono riconducibili (ad esempio, parola programmazione, tema informatica). Diverse parole chiave possono essere associate allo stesso tema. Gli utenti possono esprimere il loro interesse per alcuni temi in modo da ricevere notifiche quando vengono pubblicati post correlati al tema (tabella *Sottoscrizione*). $Notifica$ indica le notifiche dei post agli utenti amici e interessati. *Amici* indica la relazione di amicizia tra coppie di utenti.

Esercizio 01 *(Punti 9 D) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce l'identificativo di post commentati solo da amici dell'autore del post.*

Esercizio 02 *(Punti 9 B) Si scriva una interrogazione SQL che fornisce l'identificativo di post e il tema del post ed il post è tale che il suo testo contiene tutte le parole associate al tema.*

Esercizio 03 *(Punti 9 D) Si esprimano nel modo più opportuno i seguenti vincoli:*

1. *L'ordinamento dei commenti deve rispettare l'ordine di data e ora di pubblicazione (non può comparire prima nell'ordine ciò che è stato pubblicato dopo);*
2. *Se è presente una notifica del post ad un utente o l'utente è amico dell'autore del post o l'utente ha sottoscritto l'interesse al tema del post;*
3. *Un post viene notificato ad un utente solo una volta;*
4. *Le sottoscrizioni fanno riferimento a temi presenti e non cancellati.*

Esercizio 04 *(Punti 9 D) Si scriva una vista che per ogni utente ed ogni mese fornisca un riepilogo dell'attività dell'utente del tipo*
(Utente,anno,mese,N_post,N_nocom, N_com_nonsottoscrittori). (N_post il numero di post dell'utente nel mese, N_com_nonamici il numero di commenti a post dell'utente nel mese fatti da utenti che non sono sottoscrittori del tema del post, N_nocom numero di post senza commenti)

Basi di Dati e Sistemi Informativi I, 26 gennaio 2015

Facoltà di Scienze M.F.N., Corso di Laurea in Informatica

Si consideri il seguente schema relazionale che descrive la strutturazione di classi in un class diagram di UML.
$CLASS(\underline{Cod}, Nome, Descrizione)$
$REFINE(SuperClass, SubClass)$
$ATTRIBUTI(Class, Nome, Tipo, Posizione)$
$BASICTYPE(Nome)$
$METODO(\underline{CodM}, Class, Nome, TipoOut, Posizione)$
$PARAMETRI(CodM, Class, Nome, Tipo, Posizione)$

$CLASS$ descrive le classi. $REFINE$ descrive la relazione di specializzazione delle classi (SubClass è la specializzazione della classe SuperClass; entrambi gli attibuti sono totali). $ATTRIBUTI$ descrive gli attributi vengono associati ad una classe. Il tipo di un attributo può essere di tipo basico (contenuto nello schema $BASICTYPE$) oppure il nome di una classe. Posizione indica l'ordine in cui compare l'attributo nella classe. $METODO$ descrive i metodi associati ad una classe. Il $Tipout$ di ritorno di un metodo può essere di tipo basico (contenuto nello schema $BASICTYPE$) oppure il nome di una classe oppure la costante $Void$. Posizione indica l'ordine in cui compare il metodo nella classe. $PARAMETRI$ descrive i patrametri associati ad un metodo. Il $Tipout$ di un attributo può essere di tipo basico (contenuto nello schema $BASICTYPE$) oppure il nome di una classe. Posizione indica l'ordine in cui compare il parametro nel metodo. Si dice che due metodi hanno la stessa segnatura se hanno lo stesso nome, lo stesso dipo di ritorno e gli stessi parametri (in ogni posizione stesso nome e stesso tipo).

Esercizio 01 *(7 punti, II prova 0) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce codice e nome delle classi che prevedono solo metodi senza parametri.*

Esercizio 02 *(7 punti, II prova 0) Si scriva una vista SQL che restituisca per ogni classe il numero di attributi della classe di tipo BASICTYPE, il numero di metodi della classe cha hanno solo parametri di tipo BASICTYPE e il numero di classi che la specializzano.*

Esercizio 03 *(8 punti, II prova 12) Si scriva un metodo PLSQL che riceve in ingresso il codice di una classe e il codice di un metodo e che: se in una superclasse della classe data si trova un metodo con la stessa segnatura (del metodo dato in parametro) restituisce il codice del medodo trovato; altrimenti restituisce -1.*

Esercizio 04 *(8 punti, II prova 12) Utilizzando SQL dinamico, si scriva una funzione PLSQL che riceve in ingresso una sequenza di parole separate dal carattere < (ad esempio $< parola1 < parola2 < \ldots$) La funzione restituisce una stringa contenente i codici delle classi separati da < che contengono almeno una delle parole della lista di input nella loro descrizione.*

Esercizio 05 *(6 punti, II prova 10) Si consideri la relazione $R(A, B, C, D, E)$ che soddisfa le seguenti dipendenze logiche:*

- $A, B \rightarrow C$;
- $B, C \rightarrow D$;
- $B \rightarrow E$;
- $C \rightarrow A$;

Trovare le chiavi. Dire se R è in terza forma normale e se è in forma normale di Boyce-Code. Se la tabella non è in terza forma normale scomporla applicando l'algoritmo di normalizzazione.

Basi di Dati e Sistemi Informativi I, 13 febbraio 2015

DIETI, Corso di Laurea in Informatica

Si consideri il seguente schema relazionale che descrive la strutturazione di classi in un class diagram di UML.
$CLASS(\underline{Cod}, Nome, Descrizione)$
$REFINE(SuperClass, SubClass)$
$ATTRIBUTI(Class, Nome, Tipo, Posizione)$
$BASICTYPE(Nome)$
$METODO(\underline{CodM}, Class, Nome, TipoOut, Posizione)$
$PARAMETRI(CodM, Class, Nome, Tipo, Posizione)$

$CLASS$ descrive le classi. $REFINE$ descrive la relazione di specializzazione delle classi (SubClass è la specializzazione della classe SuperClass; entrambi gli attibuti sono totali). $ATTRIBUTI$ descrive gli attributi vengono associati ad una classe. Il tipo di un attributo può essere di tipo basico (contenuto nello schema $BASICTYPE$) oppure il nome di una classe. Posizione indica l'ordine in cui compare l'attributo nella classe. $METODO$ descrive i metodi associati ad una classe. Il $Tipout$ di ritorno di un metodo può essere di tipo basico (contenuto nello schema $BASICTYPE$) oppure il nome di una classe oppure la costante $Void$. Posizione indica l'ordine in cui compare il metodo nella classe. $PARAMETRI$ descrive i patrametri associati ad un metodo. Il $Tipout$ di un attributo può essere di tipo basico (contenuto nello schema $BASICTYPE$) oppure il nome di una classe. Posizione indica l'ordine in cui compare il parametro nel metodo.

Esercizio 01 *(7 punti) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce per ogni classe, il codice della classe, il nome della classe, il numero di attributi di tipo Basictype, il mumero di attributi non Basictype, il numero di metodi senza parametri.*

Esercizio 02 *(8 punti) Si scriva una interrogazione SQL che restituisca delle terne, in cui le prime due componenti sono nomi diversi di classi e la terza componente è il nome di un metodo. La condizione che la terna deve soddisfare è che in ciascuna delle due classi vi sia un metodo avente quel nome e che i due metodi abbiano la medesima segnatura (i codici sono comunque distinti). Due metodi hanno la stessa segnatura se hanno lo stesso nome, lo stesso dipo di ritorno e gli stessi parametri (in ogni posizione stesso nome e stesso tipo).*

Esercizio 03 *(6 punti) Si assuma che nella tabella BASICTYPE sia contenuto il valore Undefined. Si scriva un trigger che viene attivato quando viene cancellata una classe. Il suo effetto è quello di sostituire preventivamente, in tutti gli attributi, i parametri e il valore di ritorno dei metodi che avevano come tipo associato il nome della classe da cancellare, il nome della classe che si deve cancellare con il valore basicty Undefined.*

Esercizio 04 *(9 punti) Si scriva un metodo PLSQL che riceve in ingresso il codice di una classe e che restituisce una stringa contenente la segnatura di tutti i metodi visibili nella classe (la segnatura ha la forma nomemetodo(par1 tipo1, ..., parn tipon):tiporitorno). Si ricorda che i metodi visibili sono quelli presenti nella classe e in tutte le sue superclassi.*

Esercizio 05 *(6 punti) Si consideri la relazione $R(A, B, C, D, E)$ che soddisfa le seguenti dipendenze funzionali:*

- $A, B \rightarrow C$;
- $B, C \rightarrow D$;
- $B \rightarrow E$;
- $C \rightarrow A$;

Trovare le chiavi. Dire se R è in terza forma normale e se è in forma normale di Boyce-Code. Se la tabella non è in terza forma normale scomporla applicando l'algoritmo di normalizzazione.

Basi di Dati e Sistemi Informativi I, 27 marzo 2015

Adriano Peron

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: adrperon@unina.it

Si consideri il seguente schema relazionale che descrive un frammento della base di dati per gestire i prestiti in una biblioteca. Di un libro (descrizione del libro) possono esserci più copie fisiche (esemplari). L'attributo booleano Prestito indica se l'esemplare può essere prestato. L'attributo booleano Consultazione indica se l'esemplare può essere consultato in loco. Il prestito riguarda le copie fisiche, la prenotazione riguarda il libro: quando un esemplare del libro è disponibile si può effettuare il prestito. Ogni utente ha un profilo che regola la durata dei prestiti e il massimo numero di esemplari che può avere in prestito. Un prestito ha una data di effettuazione, una data di scadenza e una data di restituzione (che può essere NULL se il libro non è stato ancora restituito), una data di sollecito (che può essere NULL se il prestito non è ancora scaduto.)

$LIBRO(\underline{ISBN}, Titolo, Editore, Anno)$
$ESEMPLARE(ISBN, \underline{CodiceBarre}, Collocazione, Prestito, Consultazione)$
$UTENTE(\underline{CF}, CodProfilo, Nome, Cognome, DataN)$
$PROFILO(\underline{CodProfilo}, MaxDurata, MaxPrestito)$
$PRESTITI(\underline{CodPrestito}, CodiceBarre, Utente, Data, Scadenza, Restituzione, Sollecito)$
$PRENOTAZIONE(\underline{CodPrenotazione}, ISBN, Utente, Data)$

Esercizio 01 *(Punti 7) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce l'ISBN del libro che ha avuto nell'anno 2014 il maggior numero di prestiti (si intente per numero di prestiti di un libro il numero di prestiti di tutti i suoi esemplari).*

Esercizio 02 *(Punti 8) Si scriva una interrogazione in SQL che restituisca coppie di utenti che in ogni anno hanno preso in prestito lo stesso numero di libri.*

Esercizio 03 *(Punti 8) Si implementino nel modo più adeguato i seguenti vincoli:*

1. *Un utente non può avere più prestiti in corso (esemplari non ancora restituiti) di quanto previsto dal suo profilo.*
2. *Se è presente un sollecito la restituzione è stata fatta dopo la data di scadenza.*
3. *Quando viene aggiornato un prestito indicando una data di sollecito, tutte le prenotazioni di quell'utente vengo automaticamente cancellate.*

Esercizio 04 *(Punti 8) Si scriva una funzione che, quando viene eseguita, controlla quali sono i prestiti che hanno raggiunto la scadenza alla data dell'esecuzione. L'effetto della funzione di inserire nel campo Sollecito la data corrente. Inoltre, la funzione restituisce alla terminazione una stringa contenente, CF, Nome, Cognome, Titolo del libro per tutti i prestiti in ritardo (separati da ;) per cui stato indicato il sollecito.*

Esercizio 05 *(Punti 8) Si scriva una funzione che riceve in ingresso una stringa contenete delle parole separate tra loro dal simbolo −. Si scriva una interrogazione in SQL dinamico che recupera i libri in cui almeno una delle parole della stringa compare nel titolo. La funzione restituisce una stringa contenente i titoli dei libri recuperati separati dal simbolo −.*

Basi di Dati e Sistemi Informativi I, 27 marzo 2015

Adriano Peron

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: adrperon@unina.it

Si consideri il seguente schema relazionale che descrive un frammento della base di dati per gestire il log di operazioni su un file system organizzato tradizionalmente in una struttura ad albero di file e cartelle. Agli utenti sono garantiti dei diritti di operazione sui file di una cartella. I diritti di accesso sono memorizzati nella tabella $DIRITTI$. Le operazioni possibili sono lettura (R), scrittura (W), cancellazione (D), aggiunta (I). Avere un diritto sulla cartella equivale ad avere quel diritto su tutti i file della cartella (ma non sulle cartelle eventualmente contenute). Un utente ha il diritto di oprazione su un file solo se nella tabella è presente un diritto per quella operazione legato alla cartella contenente. Quando un utente effettua una operazione su un file, l'operazione viene registrata nella tabella LOG con un timestamp che indica la tempistica dell'operazione.

$FOLDER(\underline{CodF}, Nome, Path, Dimensione, FolderContenente)$
$FILE(\underline{CodF}, Nome, Folder, Path, DataC, DataM, Dimensione)$
$UTENTE(\underline{CodU}, Nome, Cognome, DataN)$
$DIRITTI(Folder, Utente, Operazione)$
$LOG(\underline{CodOp}, Utente, Operazione, File, Time)$

Esercizio 01 *(Punti 7) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce il nome e il pathname del file sul quale nel corso dell'anno 2015 sono state fatte il maggior numero di modifiche in scrittura (Si usi la funzione YEAR per estrarre l'anno dal timestamp).*

Esercizio 02 *(Punti 8) Si scriva una interrogazione in SQL che restituisca terne costituite da identificativo di folder, identificativo di utente, operazione tali per cui l'utente ha diritto all'operazione sul folder e su tutti i folder in esso immediatamente contenuti (si deve controllare un solo livello di contenimento).*

Esercizio 03 *(Punti 9) Si implementino nel modo più adeguato i seguenti vincoli:*

1. *I file che stanno nello stesso folder devono avere nomi diversi.*
2. *Se un utente fa una operazione su un file (registrata nel LOG) l'utente deve avere il diritto per quell'operazione.*
3. *La dimensione di un folder deve essere pari alla somma delle dimensioni di tutti i file e di tutti i folder contenuti.*
4. *Quando viene inserito un file nella tabella file il suo campo path viene aggiornato dopo l'inserimento in modo che risulti essere la concatenazione del path della cartella contenente e del nome del file (col tradizionale separatore).*

Esercizio 04 *(Punti 9) Si scriva una funzione che riceve in ingresso il codice di un folder e il codice di un utente. La funzione restituisce una stringa contenente il path del file, il timestamp dell'operazione e il tipo di operazione per tutte le operazioni eseguite dall'utente su file contenuti nell'albero del filesystem radicato sul folder passato come parametro. Per l'elaborazione si faccia uso di una struttura temporanea TMP che deve esere dichiarata all'esterno della funzione richiesta.*

Esercizio 05 *(Punti 6) Si consideri lo schema relazionale $R(A, B, C, D)$. La relazione gode delle seguenti dipendenze logiche:* $A, B \Longrightarrow C$
$D \Longrightarrow C$
$A \Longrightarrow D$
$D, C \Longrightarrow A$.

Dire quali dipendenze dello schema soddisfano la terza forma normale e quali la forma normale di Boyce-Codd. Ridurre lo schema in forma normale di Boyce-Codd.

Basi di Dati e Sistemi Informativi I, 25 gennaio 2016 - Completo

Adriano Peron

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: adrperon@unina.it

Si consideri il seguente schema relazionale che descrive un frammento della base di dati per gestire codice Java e attività di test sul codice. La tabella $CLASSE$ descrive le classe ed in particolare il path del file che contiene il codice e la riga iniziale e finale delle definizione della classe nel file. Ad una classe sono associati un insieme di metodi descritti nella tabella $METODI$ come per le classi viene dato il path del file che contiene il codice del metodo e la riga iniziale e finale delle istruzioni del metodo. I test per il codice sono descritti nella tabella $TEST$: anche in questo caso dato il path del file che contiene il codice del test (è esso stesso un programma) del metodo e la riga iniziale e finale delle istruzioni del test. Le esecuzioni del test su codice vengono descritte nella tabella $TESTEXEC$: si indica la data di esecuzione di un test sul codice e l'esito del test ('passato','fallito'). Nella tabella $COPERTURAM$ si memorizza informazione sull'invocazione dei metodi durante l'esecuzione di un test (copertura del metodo). Se nel corso dell'esecuzione di un test un metodo viene invocato, si memorizza una riga in corrispondeza nella tabella. Nella tabella $COPERTURAC$ si memorizza informazione sulla copertura della classe. Una classe si considera coperta in una data quando esistono coperture di tutti i suoi metodi in quella data.

$CLASSE(\underline{CodC}, Nome, Path, RigaI, RigaF)$
$METODO(\underline{CodM}, Nome, Path, , RigaI, RigaF, Segnatura, CodC)$
$TEST(\underline{CodT}, Nome, Path, , RigaI, RigaF)$
$TESTEXEC(\underline{CodE}, CodT, Data, Esito)$
$COPERTURAM(CodE, CodM)$
$COPERTURAC(CodC, Data)$

Esercizio 01 *(Punti 7) Si scriva una interrogazione in algebra relazionale che, se valutata, dia il nome e il codice delle classi che che hanno tutti i loro metodi coperti da esecuzioni di casi di test.*

Esercizio 02 *(Punti 8) Si definisca una vista che per ogni caso di test riporti il numero delle esecuzioni del test e il numero medio di metodi coperti nelle diverse esecuzioni.*

Esercizio 03 *(Punti 8) Si scriva un trigger che viene attivato all'inserimento della copertura di un metodo nella tabella $COPERTURAM$. Il trigger controlla se esiste già una copertura della classe del metodo. Se la copertura esiste già viene aggiornata la data di copertura della classe con la data di copertura del metodo. Se la copertura della classe non esiste già si verifica se tutti i metodi della classe sono stati coperti. In caso positivo si aggiunge alla tabella $COPERTURAC$ una riga per la copertura della classe.*

Esercizio 04 *(Punti 9) Si scriva una funzione SQL che riceve in ingresso il codice di una classe e che restituisce in uscita una stringa contenente i codici dei metodi di quella classe (separati) da virgole che NON sono mai stati coperti dall'esecuzione di test.*

Basi di Dati e Sistemi Informativi I, 25 gennaio 2016 - II prova B

Adriano Peron

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: adrperon@unina.it

Si consideri il seguente schema relazionale che descrive un frammento della base di dati per gestire progetti Java e le successive versioni dei progetti. La tabella $CLASSE$ descrive le classe ed in particolare il progetto a cui appartiene e il path del file che contiene il codice e la riga iniziale e finale delle definizione della classe nel file. Ad una classe sono associati un insieme di metodi descritti nella tabella $METODI$ come per le classi viene dato il path del file che contiene il codice del metodo e la riga iniziale e finale delle istruzioni del metodo. Il progetto ha diverse versioni successive. In particolare, i metodi delle classi possono avere diverse versioni indicate dal numero $NumVers$ della tabella $VERSIONEM$. L'attributo stato della versione ha possibili valori 'Updated' (il metodo viene modificato rispetto alla versione precedente), 'Deleted' (il metodo viene cancellato rispetto alla versione precedente), 'Added' (il metodo viene cancellato rispetto alla versione precedente), 'Unchanged' (rimane invariato rispetto alla versione precedente). Per ciascuna versione di un metodo si indica anche l'autore del metodo o della sua variazione. La tabella $VERSIONEC$ riporta lo stesso genere di informazioni per le classi.

$CLASSE(\underline{CodC}, Progetto, Nome, Path, RigaI, RigaF)$
$METODO(\underline{CodM}, Nome, Path, RigaI, RigaF, Segnatura, CodC)$
$VERSIONEM(NumVers, CodC, CodM, Data, codA, Stato)$
$VERSIONEC(NumVers, CodC, Data, codA, Stato)$
$AUTORE(\underline{CodA}, Nome, Cognome, Ruolo)$

Esercizio 01 *(Punti 10) Si definisca una vista che per ogni progetto e per ogni versione del progetto riporta il numero delle classi e il numero medio di metodi cambiati per classe.*

Esercizio 02 *(Punti 10) Si scriva un trigger che viene attivato all'inserimento nella tabella $VERSIONEC$. Quando viene inserita la nuova versione per una classe se la classe era presente nella versione precedente (si assuma che la versione sia un numerico) per tutti i metodi della versione precedente viene creata una copia nella tavella $VERSIONEM$ con stato 'Unchanged' nella nuova versione. Quando un metodo della versione della classe viene cambiato da 'Unchanged' a 'Changed' (o quando un metodo viene inserito con stato 'Added' o cancellato con stato 'Deleted') lo stato della classe viene aggiornato a 'Changed' aggiornando la data e l'autore con i corrispondenti valori del cambiamento del metodo.*

Esercizio 03 *(Punti 10) Si scriva una funzione SQL che riceve in ingresso il codice di una classe e che restituisce in uscita una stringa contenente i codici dei metodi di quella classe (separati da virgole) che sono associati all'ultima versione non cancellata di quella classe*

Esercizio 04 *(Facoltativo, Punti 8) Si scriva una funzione SQL che riceve in ingresso una lista di cofici di autori separati dal carattere separatore '@' e utilizzando SQL DINAMICO restituisce una lista di codici di metodi (senza duplicati) scritti dagli autori.*

Basi di Dati e Sistemi Informativi I, 25 gennaio 2016 - II Prova A

Adriano Peron

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: adrperon@unina.it

Si consideri il seguente schema relazionale che descrive un frammento della base di dati per gestire codice Java e attività di test sul codice. La tabella $CLASSE$ descrive le classe ed in particolare il path del file che contiene il codice e la riga iniziale e finale delle definizione della classe nel file. Ad una classe sono associati un insieme di metodi descritti nella tabella $METODI$ come per le classi viene dato il path del file che contiene il codice del metodo e la riga iniziale e finale delle istruzioni del metodo. I test per il codice sono descritti nella tabella $TEST$: anche in questo caso dato il path del file che contiene il codice del test (è esso stesso un programma) del metodo e la riga iniziale e finale delle istruzioni del test. Le esecuzioni del test su codice vengono descritte nella tabella $TESTEXEC$: si indica la data di esecuzione di un test sul codice e l'esito del test ('passato','fallito'). Nella tabella $COPERTURAM$ si memorizza informazione sull'invocazione dei metodi durante l'esecuzione di un test (copertura del metodo). Se nel corso dell'esecuzione di un test un metodo viene invocato, si memorizza una riga in corrispondeza nella tabella. Nella tabella $COPERTURAC$ si memorizza informazione sulla copertura della classe. Una classe si considera coperta in una data quando esistono coperture di tutti i suoi metodi in quella data.

$CLASSE(\underline{CodC}, Nome, Path, RigaI, RigaF)$
$METODO(\underline{CodM}, Nome, Path, , RigaI, RigaF, Segnatura, CodC)$
$TEST(\underline{CodT}, Nome, Path, , RigaI, RigaF)$
$TESTEXEC(\underline{CodE}, CodT, Data, Esito)$
$COPERTURAM(CodE, CodM)$
$COPERTURAC(CodC, Data)$

Esercizio 01 *(Punti 10) Si definisca una vista che per ogni caso di test riporti il numero delle esecuzioni del test, il numero di metodi coperti nelle diverse esecuzioni e il numero di metodi MAI coperti nelle diverse esecuzioni.*

Esercizio 02 *(Punti 10) Si scriva un trigger che viene attivato all'inserimento della copertura di un metodo nella tabella $COPERTURAM$. Il trigger controlla se esiste già una copertura della classe del metodo. Se la copertura esiste già viene aggiornata la data di copertura della classe con la data di copertura del metodo. Se la copertura della classe non esiste già si verifica se tutti i metodi della classe sono stati coperti. In caso positivo si aggiunge alla tabella $COPERTURAC$ una riga per la copertura della classe.*

Esercizio 03 *(Punti 10) Si scriva una funzione SQL che riceve in ingresso il codice di una classe e che restituisce in uscita una stringa contenente i codici dei metodi di quella classe (separati) da virgole che sono stati coperti dall'esecuzione di test. Se più test coprono lo stesso metodo va riportato nell'elenco solo il test che ha fornito la copertura più recentemente*

Esercizio 04 *(Facoltativo, Punti 8) Si scriva una funzione SQL che riceve in ingresso una lista di cofici di test separati dal carattere separatore '@' e utilizzando SQL DINAMICO restituisce una lista di codici di metodi (senza duplicati) coperti dai test passati in input.*

Basi di Dati e Sistemi Informativi I, 17 febbraio 2016

Adriano Peron

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: adrperon@unina.it

Si consideri il seguente schema relazionale che descrive un frammento della base di dati per gestire progetti Java e le successive versioni dei progetti. La tabella $CLASSE$ descrive le classe ed in particolare il progetto a cui appartiene e il path del file che contiene il codice e la riga iniziale e finale delle definizione della classe nel file. Ad una classe sono associati un insieme di metodi descritti nella tabella $METODO$ come per le classi viene dato il path del file che contiene il codice del metodo e la riga iniziale e finale delle istruzioni del metodo. Classi e metodi sono identificati da un codice e da un nnumero di versione (un intero). L'attributo stato della versione ha possibili valori 'Updated' (il metodo viene modificato rispetto alla versione precedente), 'Deleted' (il metodo viene cancellato rispetto alla versione precedente), 'Added' (il metodo viene aggiunto rispetto alla versione precedente), 'Unchanged' (rimane invariato rispetto alla versione precedente). Per i metodi di ciascuna versione la tabell $SCRITTURA$ indica gli autori dei metodi (associazione molti a molti). I metodi possono avere delle segnalazioni (malfunzionamenti, modifiche etc.) che hanno una data di apertura e una data di chiusura (parziale). La data di chiusura se presente coincide con la data di risoluzione del problema legata alla segnalazione.

$CLASSE(\underline{CodC, Versione}, Progetto, Nome, Path, RigaI, RigaF, Stato)$
$METODO(\underline{CodM, Versione}, Progetto, Nome, Path, RigaI, RigaF, Segnatura, CodC, Stato)$
$SCRITTURA(CodM, Versione, Data, CodA)$
$SEGNALAZIONE(\underline{CodS}, DataApertura, CodM, Versione, DataChiusura, Tipo, Descrizione)$
$ASSEGNAZIONE(CodS, CodA, DataAperetura, DataChiusura, Descrizione)$
$AUTORE(\underline{CodA}, Nome, Cognome, Ruolo)$

Esercizio 01 *(Punti 8) Si scriva una espressione in algebra relazionale che se valutata fornisce il nome delle classi che hanno mantenuti invariati (unchanged) tutti i loro metodi in tutte le versioni del progetto.*

Esercizio 02 *(Punti 8) Si implementino nel modo più adeguato i seguenti vicoli*

1. *Se per una classe esiste una istanza con $Versione > 0$, devono esistere istanze della classe per tutti i valori di versione compresi tra 0 e $Versione$.*
2. *Se la segalazione riguardante un metodo è stata assegnata a un autore, questi deve essere anche l'autore del metodo.*
3. *La data di chiusura di una segnalazione se presente deve essere successiva alla data di apertura e la descrizione non deve essere nulla;*
4. *Non ci può essere più di una segnalazione aperta (DataChiusura a NULL) per un metodo.*

Esercizio 03 *(Punti 8) Si scriva un trigger che viene attivato all'inserimento di una segnalazione per un metodo in $SEGNALAZIONE$. Il trigger aggiunge nella tabella $ASSEGNAZIONE$, l'assegnazione della gestione della segnalazione all'autore del metodo (un metodo è in genere scritto da più autori) che ha meno assegnazioni di segnalazioni aperte da gestire al momento.*

Esercizio 04 *(Punti 8) Si scriva una funzione SQL che riceve in ingresso una lista di codici di classi separate dal carattere separatore '@' e utilizzando SQL DINAMICO restituisce una lista di codici e segnature di metodi delle classi passate come parametro che sono stati cancellati nell'ultima versione disponibile.*

Basi di Dati e Sistemi Informativi I, Prove scritte AA. 2002-03

Adriano Peron

Facoltà di Scienze M.F.N., Corso di Laurea in Informatica, Dipartimento di Scienze Fisiche,
Università di Napoli 'Federico II', Italy
E-mail: peron@na.infn.it

1 Prova scritta del 2-03-04

Si consideri il seguente schema relazionale per un sistema di fatturazione e gestione di magazzino:
CLIENTE(PI,NOME)
ARTICOLO(COD-A,DESCRIZIONE,PREZZO,DISPONIBILITA)
FATTURA(ID,NUMERO,ANNO,PI,PAGATA)
VOCEFATTURA(ID,COD-A,PREZZO,QUANTITA)

Ad ogni cliente associato un numero di partita IVA (PI) ed un nome, mentre ogni articolo è caratterizzato da un codice (COD-A), da una descrizione da un prezzo e dal numero di pezzi disponibile. All'atto della vendita di alcuni prodotti ad un cliente viene emessa una FATTURA caratterizzata da un numero progressivo (azzerato allinizio di ogni anno) e da un anno, dalla partita IVA del cliente, da una chiave (ID) e da un flag (PAGATA) che indica se è avvenuto il pagamento. Ogni fattura è composta da un insieme di voci, riportate nella tabella VOCEFATTURA, che specificano il codice dell' articolo, il prezzo effettivamente pagato e la quantità acquistata.

Esercizio 11 *Si scriva una espressione in algebra relazionale che, se valutata, fornisce linsieme delle terne (nome cliente, descrizione articolo, anno) per le quali larticolo non è mai stato acquistato dal cliente nellanno indicato.*

Esercizio 12 *Si scriva una query in SQL che fornisca per ogni fattura emessa: il numero progressivo, lanno, il nome del cliente e limporto della fattura*

Esercizio 13 *Si crei una vista che fornisca per ogni coppia (cliente, anno) la partita iva, il nome del cliente, anno, fatturato complessivo sviluppato dal cliente, numero totale di fatture emesse , importo pagato, importo non anora pagato, numero di fatture pagate, numero di fatture sospese.*

Esercizio 14 *Si scriva una procedura in C o Pascal che accettando in ingresso la partita iva di un cliente guidi interattivamente lutente alla emissione di una fattura. In dettaglio occorrerà creare una nuova fattura con un numero progressivo maggiore di uno del pi alto per lanno corrente, quindi iterativamente chiedere in input delle coppie [codice articolo,quantità] dei prodotti acquistati ed inserirle opportunamente nella base di dati. Si presti attenzione alla correttezza del codice articolo e alle disponibilità effettive del magazzino. Infine i stampi, a video, la fattura completa (partita iva, nome cliente, importo totale e per ogni articolo la quantità, la descrizione il prezzo unitario e quello totale).*

Esercizio 15 *Si mostri un diagramma ER che generi lo schema della base di dati in esame. Si illustrino i vincoli di integrit referenziale intercorrenti tra le varie tabelle*

2 Prova scritta del 10-02-04

Si consideri il seguente schema relazionale che descrive la composizione di piani di studio in un corso di laurea:
$ESAME(\underline{COD-E}, TITOLO, CFU, TIPO)$
$VINCOLO(\underline{COD-E, PROP})$
$ANNO(\underline{COD-E}, AC)$
$PIANIS(\underline{MATR, COD-E}, AC)$.
Lo schema $ESAME$ contiene la descrizione di un insegnamento: codice insegnamento, nome, numero di CFU e tipo ('Obbligatorio' o 'Scelta'); Lo schema $VINCOLO$ contiene la descrizione dei vincoli di propedeuticità tra esami: se un esame di codice xxxx è propedeutico ad un esame di codice yyyy, è presente nella tabella corrispondente una riga $(yyyy, xxxx)$; Lo schema $ANNO$ contiene la descrizione degli anni di corso (con valori I, II, III etc) nell'attributo AC). Se, ad esempio, un esame di codice xxxx puo essere seguito indifferentemente al II o III anno, la tabella corrispondente contiene due righe $(xxxx, II)$ e $(xxxx, III)$. Lo schema $VINCOLO$ contiene il piano di studi di ogni studente identificato dall'attributo MATR. La tabella corrispondente contiene un una riga per ogni esame (sia obbligatorio sia a scelta) presente nel piano di studi.

Esercizio 21 *Un esame si dice di base se non ha esami propedeutici. Si scriva una espressione in algebra relazionale che, se valutata, fornisce i nomi delgi esami che se hanno esami propedeutici, hanno solo esami propedeutici di base.*

Esercizio 22 *Scrivere una interrogazione SQL che fornisca titolo e anno degli esami non obbligatori che possono essere collocati solo in un anno (non c'è scelta d'anno).*

Esercizio 23 *Scrivere una vista che fornisca, per ogni studente, un riepilogo che permetta di controllare la correttezza dei piani di studio. La vista deve avere i seguenti campi: la matricola dello studente, il numero di obbligatori mancanti, il numero di esami collocati in anni sbagliati, il numero di CFU complessivo derivante da esami al al I anno, al II anno e al III anno.*

Esercizio 24 *Si scriva una procedura in C o Pascal che riceve in ingresso una matricola di uno studente e restituisce in uscita il numero degli insegnamenti presenti nel piano di studi per i quali manchino esami propedeutici. Inoltre, la procedura, per ogni insegnamento propedeutico mancante, inserisce una riga in una tabella (da supporre esistente al momento dell'esecuzione della procedura) avente schema*
$ERRATI(MATR, COD-E, COD-PROP)$ *(COD-E è il codice dell'insegnamento presente nel piano di studi e COD-PROP è il codice del suo insegnamento propedeutico mancante).*

Esercizio 25 *Si dica cosa si intende con il termine transazione e si dica quali sono i costrutti per programmare una transazione. Si descrivano inoltre brevemente le proprietà 'acide' (ACID) delle transazioni.*

3 Prova scritta del 28-09-03

Si consideri il seguente schema relazionale che descrive il trasferimento di file in un sistema di file-sharing:
$UTENTE(NOME, BANDA-IN, BANDA-OUT)$
$POSSIEDE(NOME, NOMEF)$
$DOWNLOAD(PROP, RIC, NOMEF, BANDA)$

Ogni utilizzatore ($UTENTE$) del sistema è caratterizzato da un $Nome$, da una banda in ingresso ($Banda-IN$) e da una banda in uscita ($Banda-OUT$). Ogni utente può condividere diversi file come espresso dalla tabella $POSSIEDE$, dove $Nomef$ è il nome del file e $Nome$ è il nome di un utente che lo condivide. Lo schema $DOWNLOAD$ rappresenta i trasferimenti attivi, $Prop$ è il nome dellutente da cui viene scaricato il file, $Nomef$ è il nome del file scaricato, Ric è il nome dellutente che lo sta scaricando, mentre $Banda$ è la banda impiegata per quel trasferimento.

Esercizio 31 *Si scriva un'espressione in algebra relazionale che, se valutata, fornisce i nomi di tutti i file posseduti da un solo utente (non si usino operazioni di conteggio).*

Esercizio 32 *Scrivere un'interrogazione SQL che fornisce tutte le possibili coppie* $(Usr, Nomef)$ *dove* $Nomef$ *non è posseduto da* Usr.

Esercizio 33 *Si crei una vista che fornisca per ogni utente: banda in ingresso totale, banda effettivamente impiegata da download, banda in uscita totale, banda effettivamente impiegata da upload, numero di download ed upload attivi. Si assegni all'utente USER il diritto di lettura sulla vista.*

Esercizio 34 *Si scriva una procedura in C o in PASCAL che riceve in ingresso il nome di un utente, il nome di un file, e la banda richiesta per il download. La procedura dovrà inserire una nuova riga in DOWNLOAD relativo al file e allutente indicato. Ciò dovrà essere fatto provando a scaricare il file dallutente che ha disponibile in uscita (tenedo conto degli upload correnti) la più piccola banda maggiore di quella richiesta. Nel caso nessuno abbia a disposizione una banda in uscita sufficiente, allora il download sarà effettuato presso lutente con maggior banda disponibile (ovviamente occorrerà controllare che la banda richiesta sia compatibile con la banda in ingresso disponibile dell'utente che richiede il file). In uscita la procedura restituirà il nome dell'utente da cui viene scaricato il file e la banda assegnata. Nel caso il file non sia disponibile presso nessun utente oppure sia già posseduto dall'utente indicato si restituirà una segnalazione di errore.*

Esercizio 35 *Nell'ambito del controllo di affidabilità dei sistemi per la gestione delle basi di dati si descriva l'utilità e la struttura dei file di log. Inoltre si descriva l'operazione di ripresa a caldo in seguito ad un malfunzionamento.*

Soluzione Esercizio 31
Sia PS la relazione per POSSIEDE.
Nomi di file posseduti da almeno due persone.

$$PIU \leftarrow \Pi_{Nomef}(\sigma_C(\rho_{PR1(Usr1,Nomef1)}(PS) \times PS))$$

dove C è la condizione $Usr <> Usr1 \wedge Nomef = Nomef1$.
Risultato.

$$\Pi_{Nomef}(PS) \setminus PIU$$

† **Soluzione Esercizio 32**

```
SELECT A.Usr,B.Nomef
FROM   UTENTE AS A, POSSIEDE AS B
WHERE B.Nomef NOT IN
       (SELECT Nomef
       FROM   POSSIEDE AS C
       WHERE  A.Usr = C.Usr)
```

Una soluzione alternativa basata sulle operazioni insiemistiche può essere la seguente:

```
SELECT A.Usr,B.Nomef
FROM   UTENTE AS A, POSSIEDE AS B
EXCEPT
SELECT *
FROM   POSSIEDE
```

†
Soluzione Esercizio 33
Vista per il computo dell'informazione riguardante UPLOAD per un generico utente.

```
CREATE VIEW Upload(Usr,BandaU,NU) AS
       SELECT Prop,SUM(Banda),Count(*)
       FROM   DOWNLOAD
       GROUP BY Prop
```

Vista per il computo dell'informazione riguardante DOWNLOAD per un generico utente.

```
CREATE VIEW Download(Usr,BandaD,ND) AS
       SELECT Ric,SUM(Banda),Count(*)
       FROM   DOWNLOAD
       GROUP BY Ric
```

Creazione della vista finale.

```
CREATE VIEW Conteggio(Usr, Banda-IN, BandaD, ND, Banda-OUT, BandaU, NU) AS
       SELECT Usr, Banda-IN, BandaD, ND, Banda-OUT, BandaU, NU,
       FROM   UTENTE NATURAL JOIN Download NATURAL JOIN
              Upload
```

Assegnazione dei privilegi
GRANT SELECT ON Conteggio TO USER †
Soluzione Esercizio 34
La procedura proposta, per semplicità di soluzione, non adotta una strategia di interrogazione del database efficiente (duplicazione dell'interrogazione).

```
PROCEDURE Scarica (IN Utente,File: string, Banda: integer
                   OUT Error : boolean)
```

```
VAR
EXEC SQL BEGIN DECLARE SECTION
     Ut, Ut2, Fl,Pr : string;
     Bd, Ds : integer
EXEC SQL END DECLARE SECTION

BEGIN
Ut:=Utente; Fl:= File; Bd:= Banda; Error:= False
{**Verifica che il file non sia gia' posseduto**}
EXEC SQL SELECT Usr INTO :Ut2
              FROM   POSSIEDE
              WHERE Usr = :Ut AND Nomef = :Fl;
IF SQLCODE=0 THEN Error := TRUE;
{**Verifica che il file sia disponibile con una banda sufficiente**}
IF NOT Error THEN
BEGIN
EXEC SQL DECLARE disp CURSOR FOR
              SELECT Usr, Banda-OUT - SUM(Banda) AS Libero
              FROM   UTENTE,POSSIEDE,DOWNLOAD
              WHERE UTENTE.Usr = POSSIEDE.Usr AND
                        UTENTE.Usr = DOWNLOAd.PROP AND
                        POSSIEDE.Nomef = :Fl
              GROUP BY Usr
              HAVING Libero ≥ :Bd
              ORDER BY Libero;
EXEC SQL OPEN disp;
EXEC SQL FETCH disp INTO :Pr, :Ds;
IF SQLCODE = 0 THEN
{**File trovato con banda sufficiente**}
EXEC SQL INSERT INTO DOWNLOAD VALUES (:Pr,:Ut,:Fl,:Bd)
ELSE
BEGIN
{**Cerca file con banda inferiore alla richiesta**}
EXEC SQL DECLARE disp2 CURSOR FOR
              SELECT usr, Banda-OUT - SUM(Banda) AS Libero
              FROM   UTENTE,POSSIEDE,DOWNLOAD
              WHERE UTENTE.Usr = POSSIEDE.Usr AND
                        UTENTE.Usr = DOWNLOAd.Prop AND
                        POSSIEDE.Nomef = :Fl
              GROUP BY usr
              HAVING Libero > 0
              ORDER BY Libero DESC;
EXEC SQL OPEN disp2;
EXEC SQL FETCH disp INTO :Pr, :Ds;
IF SQLCODE = 0 THEN
EXEC SQL INSERT INTO DOWNLOAD VALUES (:Pr,:Ut,:Fl,:Ds)
ELSE Error := True
END
END
```

END

†

4 Prova scritta del 8-09-03

Si consideri il seguente schema relazionale che descrive la composizione di linee ferroviarie, della composizione dei treni che effettuano i tragitti e delle prenotazioni dei posti sui treni. $LINEA(\underline{COD-L}, COD-T, ST-P, ST-A, OP, OA)$
$TRATTA(\underline{COD-L, ST-P}, ST-A, OP, OA)$
$C-TRENO(\underline{COD-T, N-VAG}, Posti, Tipo)$
$PRENOTAZIONI(\underline{Cod-P}, Data, COD-L, ST-P, ST-A, OP, OA, N-VAG, N-Posto)$

Ogni linea ferroviaria (LINEA) è indicata univocamente da un codice COD-L, da un codice del treno che effettua il servizio (COD-T), dalle stazioni di partenza e destinazione (ST-P e ST-A) e dai relativi orari di partenza ed arrivo (OA e OP). Le fermate intermedie di una linea sono descritte dallo schema TRATTA. Una tratta va intesa come una porzione della linea che non ha fermate intermedie ed è indicata dalla stazione di partenza ed arrivo (ST-P e ST-A) e dai relativi orari (OA e OP). (Una istanza di tratta è presente in TRATTA anche se la linea corrispondente è costituita da un'unica tratta.) Lo schema C-TRENO descrive la composizione dei treni (in vagoni): ogni istanza descrive un vagone che compone il treno COD-T indicandone il numero (N-VAG) ed il numero complessivo di posti disponibili nel vagone (si assume che che se N è il numero di posti complessivi i posti siano indicati dai numeri progressivi 1..N). Le prenotazioni sono descritte in $PRENOTAZIONI$. Ogni istanza di prenotazione è identificata da un codice, data, linea ferroviaria, stazione ed ora di partenza ed arrivo, numero di vagone e di posto. La prenotazione riguarda soltanto il sottoinsieme delle tratte della linea comprese tra la stazione di partenza ed arrivo (e non, dunque, necessariamente una singola tratta o l'intera linea).

Esercizio 41 *Si scriva una espressione in algebra relazionale che, se valutata, fornisce il codice di linea, la data, il numero di vagone e di posto a cui siano associate nell'intero tragitto almeno due prenotazioni distinte. (Esempio: nella stessa corsa sulla linea NA-RM-FI un posto può avere due diverse prenotazioni, una sulla tratta NA-RM e l'altra sulla tratta RM-FI). Nello svolgimento dell'esercizio NON si usino operazioni di raggruppamento.*

Esercizio 42 *Scrivere una interrogazione SQL che fornisca come risultato, codice di linea, data, numero di vagone e di posto, per posti che siano prenotati per l'intera linea (non si trascuri il caso generale in cui la prenotazione sull'intera linea risulta dal cumulo di prenotazioni distinte su parti della linea).*

Esercizio 43 *Si crei una vista che fornisca data, codice di linea, numero complessivo di posti disponibili nel treno, numero complessivo di posti per cui esiste una prenotazione almeno su una tratta (non necessariamente sull'intera linea), numero di posti per cui non esiste nessuna prenotazione, numero di posti che sono prenotati in alcune tratte ma non in tutte. Si assegni un diritto di lettura all'utente 'USER' sulla vista.*

Esercizio 44 *Si scriva una procedura in C o Pascal che riceve in ingresso i seguenti dati per effettuare una prenotazione: Codice di prenotazione,Data, Codice di linea,Stazione di partenza,Stazione di arrivo (Ora di partenza e di arrivo non vengono forniti e devono essere calcolati all'interno della procedura). La procedura prima controlla tra i posti che già sono stati prenotati per verificare se è possibile assegnare la prenotazione ad un posto già parzialmente prenotato. Se esistono posti che soddisfano questi requisiti la prenotazione viene effettuata su uno qualsiasi di essi. Se invece posti parzialmente occupati compatibili non esistono, allora si verifica se vi siano posti liberi su tutta la linea: se ve ne sono, si prenota il primo posto disponibile (ordinando i posti per numero di vagone e di posto) altrimenti si fornisce una segnalazione di indisponibilità.*

Esercizio 45 *Nell'ambito del controllo di affidabilità dei sistemi per la gestione delle basi di dati si descriva l'utilità e la struttura del file di log. In particolare si indichi a quali operazioni di sistema corrispondono i record di Dump e Checkpoint che nel file di log possono essere registrati.*

4.1 Soluzioni

(NOTA. Nella valutazione del compito, ha ottenuto pieno punteggio chi abbia svolto correttamente almeno tutti gli esercizi ad eccezione del quarto)
Soluzione Esercizio 41
Sia PR la relazione per PRENOTAZIONI.
Copia della relazione di prenotazione.

$$PR1 \leftarrow \rho_{PR1(Cod-P_1,Data_1,...,N-Vag_1,N-Posto_1)}(PR)$$

Risultato.

$$\Pi_{Cod_L,Data,N-Vag,N-Posto}(PR \bowtie_C PR_1)$$

dove C è la condizione $Cod-L = Cod-L_1 \wedge Data = Data_1 \wedge N-Vag = N-Vag_1 \wedge N-Posto = N-Posto_1 \wedge Cod-P <> Cod-P_1$ †
Soluzione Esercizio 42
L'idea adottata in questa soluzione è di elencare i posti per i quali non vi sia una tratta non coperta da prenotazione.

```
SELECT Cod-L,Data,N-Vag,N-Posto
FROM   PRENOTAZIONI AS A
WHERE NOT EXISTS
       (SELECT*
       FROM   TRATTA AS B
       WHERE A.Cod-L = B.Cod-L AND
              NOT EXISTS
              (SELECT*
              FROM   PRENOTAZIONI AS C
              WHERE A.Cod-L = C.Cod-L AND A.Data = C.Data AND
                     A.N-Vag = C.N-Vag AND A.N-Posto = C.N-Posto AND
                     C.OP ≤ B.OP AND C.OA ≥ B.OA))
```

†
Soluzione Esercizio 43
Vista per il computo del numero di posti complessivi

```
CREATE VIEW PostiTotatli(Cod-L,TotPosti) AS
        SELECT Cod-L,SUM(Posti)
        FROM   LINEA NATURAL JOIN C-TRENO
        GROUP BY Cod-L
```

Vista per il computo dei posti occupati

```
CREATE VIEW PostiOccupati(Cod-L,Data,TotOccupati) AS
        SELECT Cod-L, Data, COUNT(DISTINCT N-Vag,N-Posto)
        FROM   PRENOTAZIONE
        GROUP BY Cod-L, Data
```

Vista per il computo dei posti occupati su ogni tratta (riconducibile all'esercizio 42)

```
CREATE VIEW SempreOccupati(Cod-L,Data,TotSempre) AS
        SELECT Cod-L, Data, COUNT(DISTINCT N-Vag,N-Posto)
        FROM   *Come in esercizion 42
        WHERE *Come in esercizion 42
        GROUP BY Cod-L, Data
```

Creazione della vista finale.

```
CREATE VIEW Conteggio(Cod-L,Data,TotPosti,Occupati,Liberi,Parziali) AS
        SELECT Cod-L, Data, TotPosti,TotOccupati,
                TotPosti-TotOccupati,TotOccupati-TotSempre
        FROM   PostiTotali NATURAL JOIN PostiOccupati NATURAL JOIN
SempreOccupati
```

Assegnazione dei privilegi GRANT SELECT ON Conteggio TO USER †

Soluzione Esercizio 44

```
PROCEDURE Prenota (IN CodiceP,CodiceL,DataP,SP,SA : string,
                        OUT Prenotato : boolean)

VAR
EXEC SQL BEGIN DECLARE SECTION
    OraP,OraA : string;
    CodPren,CodLin,StazP,StazA,DPart : string;
    Vagone,Posto : integer
EXEC SQL END DECLARE SECTION

BEGIN
CodPren:=CodiceP; CodLin:= CodiceL; Dpart:=DataP
StazP:=SP; StazA:=SA;
Prenotato:=False;
{**Calcolo dell'ora di partenza ed arrivo**}
EXEC SQL SELECT OP INTO :OraP
            FROM   TRATTA
            WHERE Cod-L = :CodLin AND ST-P = :StazP;
EXEC SQL SELECT OA INTO :OraA
            FROM   TRATTA
```

```
        WHERE Cod-L = :CodLin AND ST-A = :StazA;
{**Definizione del cursore per l'insieme dei posti gia' prenotati che consentono l'aggiunta
della prenotazione richiesta **}
EXEC SQL DECLARE occ CURSOR FOR
        SELECT N-Vag, N-Posto
        FROM   PRENOTAZIONI AS A
        WHERE Cod-L = :CodLin AND Data = :Dpart AND NOT EXISTS
                (SELECT *
                FROM PRENOTAZIONI AS B
                WHERE A.Data = B.Data AND A.Cod-L = B.Cod-L AND
                        A.N-VAg = B.N-Vag AND A.N-Posto = B.N-Posto AND
                        ((OP ≤ :OraP AND OA > :OraP) OR
                        (OP ≥ :OraP AND OP < :OraA) ))
        ORDER BY N-Vag, N-Posto;
{**Definizione del cursore per l'insieme dei posti liberi su cui inserire la prenotazione.
Un posto libero e' il successivo del posto di numero massimo occupato in un vagone **}
EXEC SQL DECLARE lib CURSOR FOR
        SELECT = N-Vag, MAX(N-Posto)+1
        FROM   PRENOTAZIONI NATURAL JOIN LINEE NATURAL JOIN C-TRENO
        WHERE Cod-L = :CodLin AND Data = :Dpart
        GROUP BY N-Vag
        HAVING MAX(N-Posto)< C-TRENO.Posti
        ORDER BY N-Vag
EXEC SQL OPEN occ;
{**Se il cursore occ contiene almeno una riga esiste un posto prenotato a cui e' possibile
aggiungere la prenotazione corrente **}
EXEC SQL FETCH occ INTO :Vagone, :Posto;
IF SQLCODE=0 THEN
        BEGIN
        Prenotato := TRUE;
        INSERT INTO PRENOTAZIONI VALUES (:CodPren,:Dpart,:CodLin,
                :StazP,:StazA,:OraP,:OraA,:Vagone,:Posto)
        END
ELSE
        BEGIN
        EXEC SQL OPEN lib;
        IF SQLCODE=0 THEN
                BEGIN
                Prenotato := TRUE;
                INSERT INTO PRENOTAZIONI VALUES (:CodPren,:Dpart,:CodLin,
                        :StazP,:StazA,:OraP,:OraA,:Vagone,:Posto)
                END
        END
END
```

†

5 Prova scritta del 22-07-03

Si consideri il seguente schema relazionale che descrive la composizione di tratte ferroviarie:
$LINEA(\underline{COD-L}, NomeTreno, CittaP, CittaA, OraP, OraA, km)$
$TRATTA(\underline{COD-L, CittaP}, CittaA, OraP, OraA, Km)$
$GEO(Citta, Provincia, Regione)$.
Lo schema $LINEA$ contiene la descrizione del tragitto complessivo del treno: città di partenza e arrivo, ora di partenza e arrivo, lunghezza complessiva. Lo schema $TRATTA$ contiene la descrizione delle singole tratte che costituisco una linea (sottopercorso senza fermate intermedie): città di partenza e arrivo, ora di partenza e arrivo, lunghezza della tratta.

Esercizio 51 *Si scriva una espressione in algebra relazionale che, se valutata, fornisce l'insieme dei nomi dei treni regionali, cioè dei treni su linee con tutte le fermate nella medesima regione.*

Esercizio 52 *Scrivere una vista che fornisca l'elenco delle coppie dei codici delle linee (cioè uno schema del tipo $INTERSECT(Cod-L1, Cod-L2)$) che contenga le coppie dei codici delle linee che hanno una stazione di coincidenza, cioè di arrivo per una linea e partenza per l'altra (considerare anche le stazioni interne alle linee).*
Sfruttare la vista ottenuta per fornire l'elenco delle coppie delle città che non sono collegate da una linea diretta ma che sono raggiungibili mediante un unico cambio di treno.

Esercizio 53 *La base di dati sopra descritta è accessibile a due categorie di utenti: operatori e generici. Un operatore ha accesso in lettura a tutte le informazioni e può solo fare variazioni sui campi degli orari delle tabelle. L'utente generico ha accesso in lettura ad una tabella dei treni regionali (di schema $LINEAREG(Regione, \underline{COD-L}, NomeTreno, CittaP, CittaA, OraP, OraA, km)$) ed alla sottotabella di $TRATTE$ che si riferisce ai treni regionali.*

Esercizio 54 *Si immagini che oltre alle tabelle sopra descritte, nella base di dati vi sia una tabella $COPPIE(\underline{COD-L, CittaP, CittaA}, OraP, OraA, Km)$*
che mantiene informazione su coppie di città partenza-destinazione che stanno su una medesima linea e che a differenza di $TRATTE$ possono avere tra la partenza e la destinazione delle altre tappe intermedie (km in questo caso riporta la distanza complessiva tra la partenza e la destinazione).
Si scriva una procedura in C o Pascal che riceve in ingresso il codice di una linea e che ha come effetto l'inserimento di tutte le coppie partenza-destinazione nella tabella $COPPIE$ per la linea data. La distanza complessiva tra la partenza e la destinazione è la somma delle lunghezze delle singole tratte. (Suggerimento: si interroghi la base di dati per le coppie ammissibili e per ciascuna coppia si ottenga mediante altra interrogazione il computo dei km complessivi.)

Esercizio 55 *Si descriva la tecnica del timestamping per la gestione dell'accesso concorrente alle basi di dati. Si forniscano poi, come esemplificazione, due sequenze di letture e scrittura operate (ciascuna) da due transazioni $T1$ e $T2$. La prima sequenza di operazioni sia costruita in modo tale da rispettare la politica di timestamping (nessuna lettura o scrittura della sequanza viene cancellata). La seconda sequenza sia costruita in modo da violare la politica di timestamping.*

5.1 Soluzioni

Soluzione Esercizio 51
Siano LN, TT e GO la relazione per LINEA, TRATTA e GEO, rispettivamente.
Tratte con indicazione delle regioni di partenza e arrivo.

$$TT - REG \leftarrow \Pi_{Cod-L,Reg-P,Reg-A}(((TT \bowtie \rho_{GOP(CittaP,PrP,Reg-P)}(GO)) \bowtie \bowtie \rho_{GOA(CittaA,PrA,Reg-A)}(GO))$$

Codici di linea che non hanno tratte interregionali.

$$LN - REG \leftarrow \Pi_{Cod_L}(LN) \setminus \Pi_{Cod-L}(\sigma_{Reg-P \neq Reg-A}(TT - REG))$$

Soluzione

$$\Pi_{NomeTreno}(LN - REG \bowtie LN)$$

† **Soluzione Esercizio 52**
La soluzione proposta adotta l'ipotesi che per ogni linea in una direzione esista anche la linea corrispondente e similare (stesse tratte) nella direzione opposta.
Creazione della vista Intersect

```
CREATE VIEW Intersect(Cod-L1,Cod-L2) AS
        SELECT A.Cod-l,B.Cod-l
        FROM   TRATTA A, TRATTA B
        WHERE A.Cod-l <> B.Cod-l AND
                (A.CittaP=B.CittaA OR A.CittaA=B.CittaP)

SELECT A.CittaP,B.CittaA
FROM   TRATTA A, TRATTA B, Intersect
WHERE A.Cod-L=Cod-L1 AND B.Cod-L= Cod-L2 AND
        A.CittaP<>B.CittaA
EXCEPT
SELECT A.CittaP,B.CittaA
FROM   TRATTA A, TRATTA B
WHERE A.Cod-L=B.Cod-L
```

†
Soluzione Esercizio 53
Per l'operatore non vanno create viste e i privilegi vanno assegnati direttamente sulle tabelle. GRANT SELECT ON LINEA TO Operatore
GRANT SELECT ON TRATTA TO Operatore
GRANT SELECT ON GEO TO Operatore
GRANT UPDATE(OraP,OraA) ON LINEA TO Operatore
GRANT UPDATE(OraP,OraA) ON TRATTA TO Operatore
Creazione della vista delle linee regionali per l'utente generico.

```
CREATE VIEW LineaRegionale AS
        SELECT Regione, A.*
        FROM   (LINEA AS A) JOIN GEO ON (CittaP=Citta)
        WHERE A.Cod-L NOT IN
                SELECT Cod-L
```

```
        FROM    (TRATTA JOIN (GEO AS K) ON CittaP = K.Citta)
                JOIN GEO AS H ON CittaA = H.Citta
        WHERE H.Regione<>K.Regione
```

Creazione della vista delle tratte regionali per l'utente generico.

```
CREATE VIEW TrattaRegionale AS
        SELECT *
        FROM    TRATTA
        WHERE Cod-L IN (SELECT Cod-L FROM LineaRegionale)
```

Assegnazione dei privilegi a Generico GRANT SELECT ON LineaRegionale TO Generico
GRANT SELECT ON TrattaRegionale TO Generico †

Soluzione Esercizio 54

```
PROCEDURE Caricacoppie (IN Cod : string)
VAR
EXEC SQL BEGIN DECLARE SECTION
    OraPartenza,OraArrivo : date;
    Codice,CittaPartenza,CittaArrivo : string;
    Totale : integer
EXEC SQL END DECLARE SECTION

BEGIN
Codice:=Cod;
EXEC SQL DECLARE cc CURSOR FOR
            SELECT A.CittaP, B.CittaA, A.OraP, B.OraA
            FROM    TRATTA AS A JOIN TRATTA AS B ON A.Cod-L = B.od-L
            WHERE A.Cod-L = :Codice AND A.OraP < B.OraA;
EXEC SQL OPEN cc;
EXEC SQL FETCH cc INTO :Partenza, :Arrivo, :OraPartenza, :OraArrivo;
WHILE SQLCODE=0 DO
            BEGIN
            EXEC SQL SELECT SUM(km) INTO :Totale
                        FROM TRATTA
                        WHERE Cod-L = :Codice AND
                                    OraP >= :OraPartenza AND OraA <= :OraArrivo;
            EXEC SQL INSERT INTO Coppie VALUES
                        (:Codice,:Partenza, :Arrivo, :OraPartenza, :OraArrivo, :Totale);
            EXEC SQL FETCH cc INTO :Partenza, :Arrivo, :OraPartenza, :OraArrivo;
            END;
END

†
```

6 Prova scritta del 25-06-2003

Si consideri il seguente schema relazionale che descrive il modo in cui alcuni pezzi meccanici semplici sono integrati tra loro per comporre nuovi pezzi e che tiene informazione sui produttori dei pezzi:
$PEZZO(\underline{COD-P}, TIPO)$
$PRODUTTORE(\underline{MARCA}, PI, INDIRIZZO)$
$COMPONE(\underline{COD-PARTE, COMPOSTO}, QUANTITA)$
$CATALOGO(\underline{COD-P, MARCA}, COSTO)$.
Lo schema $COMPONE$ indica se un pezzo (COD-PARTE) è parte costitutiva di di un pezzo composto (COMPOSTO) e in quale quantità. Lo schema $CATALOGO$ indica quali produttori forniscono i pezzi e a quale prezzo.

Esercizio 61 *Si scriva una espressione in algebra relazionale che, se valutata, fornisce l'elenco del TIPO dei pezzi composti da soli pezzi di base (cioè non scomponibili in altri pezzi).*

Esercizio 62 *Fornire mediante una interrogazione SQL l'elenco dei produttori (MARCHE) e la loro Partita Iva (PI) in grado di fornire almeno tutti i pezzi forniti dal produttore ALLMEC.*

Esercizio 63 *Si immagini che vi siano le seguenti categorie di utenti: Amministrazione, Segreteria, Utente. Si progetti in SQL un insieme di viste (con relativi diritti) sapendo che*

- *l'utente Amministrazione ha accesso in lettura e scrittura a tutte le tabelle;*
- *l'utente Segreteria puó accedere in lettura a tutte le tabelle, inserire dati nelle tabelle PEZZI, PRODUZIONE e CATALOGO, variare solo il costo nella tabella CATALOGO;*
- *l'utente Utente può accedere in lettura al catalogo dei pezzi di base limitatamente ai campi MARCA, COSTO, TIPO e ad un catalogo riassuntivo dei pezzi di base dove ad ogni tipologia di pezzo viene associata la marca che garantisce il prezzo minimo, il prezzo minimo, la marca che grantisce il prezzo massimo ed il prezzo massimo.*

Esercizio 64 *Si integri lo schema relazionale sopra descritto con due nuovi schemi di relazioni che fungono da deposito storico per i pezzi fuori commercio e per il catalogo fuori commercio:* $STORICOPEZZO(\underline{COD-P}, TIPO)$ *e*
$STORICOCATALOGO(\underline{COD-P, MARCA}, COSTO)$.
Si definisca una procedura in C o PASCAL che riceve in ingresso il codice di un pezzo e sortisce come risultato il trasferimento delle righe del catalogo che hanno quel codice dalle tabelle $PEZZO$ *e* $CATALOGO$ *alle tabelle* $STORICOPEZZO$ *e* $STORICOCATALOGO$*, rispettivamente. Per organizzare le operazioni si tenga conto che vi è un vincolo di integrità referenziale tra la tabella* $CATALOGO$ *e la tabella PEZZI che impedisce la violazione del vincolo in cancellazione.*

Esercizio 65 *Si definisca la nozione di* indice primario *e se ne descriva brevemente la struttura.*

7 Elaborato di prova, Maggio 2003

Negli esercizi di seguito elencati si faccia riferimento agli schemi relazionali di seguito riportati:

$$Paziente(\underline{CF}, Nome, Cognome, Indirizzo, Telefono)$$

$$Ricovero(\underline{CodR}, DataI, DataF, CF)$$

$$Esame(\underline{CodE, CodR, data}, esito)$$

$$TipoEsame(\underline{CodE}, Descrizione, Ticket)$$

dove, CF indica il Codice Fiscale del paziente; $CodR$, $DataI$ e $DataF$ indicano il codice identificativo la data di inizio e di fine, rispettivamente, del ricovero; $CodE$ indica il codice identificativo dell'esame.

Esercizio 71 *Scrivere una espressione dell'algebra relazionale che, valutata, fornisca Nome, Cognome dei pazienti la Descrizione degli esami sostenuti almeno due volte nell'ambito di uno stesso ricovero.*

Esercizio 72 *Scrivere una interrogazione SQL che fornisca Nome e Cognome dei pazienti e l'ammontare totale dei tiket relativi agli esami a cui si sono sottoposti nel loro ultimo ricovero.*

Esercizio 73 *Si immagini che la base di dati sia accessibile a tre categorie distinte di utenti: Medico, Amministratore, Esterno. Gli utenti hanno i seguenti privilegi di accesso: l'utente Medico ha accesso in lettura e scrittura (r/w) a tutti i dati; l'utente Amministratore ha acecsso r/w ai dati di TipoEsame, in lettura ai dati di Paziente e Ricovero, ed ha accesso in lettura solo alla data e al costo (ticket) degli Esami (non deve conoscere la tipologia dell'esame e l'esito); l'utente Esterno ha accesso in lettura soltanto ad un elenco con nome e cognome degli utenti attualmente ricoverati.*
Definire un insieme di viste e privilegi per le tre tipologie di utenti descritti.

Esercizio 74 *Scrivere una procedura che riceve in ingresso una data e, dopo aver recuperato nella base di dati tutti i pazienti ancora degenti a partire da quella data, restituisce in uscita un array di dimensioni (Mx2) inizializzatao col nome e cognome dei pazienti selezionati.*

Esercizio 75 *Si discutano le linee essenziali della tecnica del locking gerarchico.*

Basi di Dati e Sistemi Informativi I, Prove scritte AA 2003/04

Adriano Peron

Facoltà di Scienze M.F.N., Corso di Laurea in Informatica, Dipartimento di Scienze Fisiche,
Università di Napoli 'Federico II', Italy
E-mail: peron@na.infn.it

1 Prova scritta del 8-09-04

Si consideri il seguente schema relazionale per la descrizione bibliografica di articoli pubblicati su riviste scientifiche:
$RIVISTA(\underline{COD-R}, TITOLO-R, EDITORE)$
$ARTICOLO(\underline{COD-AR}, COD{-}R, TITOLO, NUMERO, ANNO, PAG{-}I, PAG{-}F)$
$AUTORI(\underline{COD-AU}, Nome, Cognome)$
$SCRIVE(COD-AR, COD-AU)$
$CITAZIONI(COD-CITATO, COD-RIF)$.
Una istanza di $ARTICOLO$ ha come attributi il codice dell'articolo $COD-AR$ (chiave) il codice della rivista, il titolo dell'articolo, il numero del fascicolo della rivista, l'anno, la pagina iniziale e finale; Un articolo può avere uno o più autori. Gli autori sono descritti dallo schema $AUTORI$ e lo schema $SCRIVE$ codifica il legame molti a molti tra autori e articoli: se xx è il codice di un autore e yy è il codice di un articolo, una istanza (yy, xx) di $SCRIVE$ indica che xx è un autore di yy. $CITAZIONI$ memorizza i riferimenti tra articoli: se (zz, yy) è una istanza di CITAZIONI, allora yy è il codice di un articolo che cita un altro articolo zz nella sua bibliografia.

Esercizio 11 *Si scriva (senza usare operazioni di conteggio) una espressione in algebra relazionale che, se valutata, fornisce nome, cognome degli autori che non abbiano mai scritto da soli un articolo (gli articoli da loro scritti hanno sempre più di un autore. Suggerimento: si trovino prima gli articoli scritti da un solo autore ...).*

Esercizio 12 *Scrivere una interrogazione SQL che fornisca titolo di rivista e titolo dell'articolo pubblicato nella rivista avente massimo numero di citazioni.*

Esercizio 13 *Si scriva una procedura* DescrizioneBibliografica *in C o Pascal che riceve in ingresso il codice di un articolo e restituisce in uscita una stringa di testo che rappresenta la descrizione bibliografica completa dell'articolo; la stringa fornita in uscita deve avere la seguente struttura: <lista cognome nome degli autori in ordine alfabetico>, <titolo articolo>,<titolo rivista>,<numero>,<anno>,<pagina iniziale>, <paginafinale>.*
Sfruttando la procedura DescrizioneBibliografica *si scriva una procedura* Bibliografia *che riceve in ingresso il codice di un articolo e restituisce in uscita una stringa di testo ottenuta concatenando la descrizione bibliografica completa di tutti gli articoli citati nell'articolo passato come parametro.*

Esercizio 14 *1. Si dica che cosa si intende con indice primario;*
2. Un indice primario è denso (giustificare la risposta)?
3. Si descrivano la struttura ed i vantaggi di un indice multilivello.

Soluzione Esercizio 11
Siano RI, ART, AU, SCR CIT le relazione per RIVISTE, ARTICOLI, AUTORI, SCRIVE e CITAZIONE rispettivamente.
Articoli scritti da più di un autore.

$$PIU \leftarrow \Pi_{COD-AR}(\rho_{SCR1(COD-AR1,COD-AU1)}(SCR) \bowtie_C SCR))$$

dove C è la condizione $COD-AR = COD-AR1 \wedge COD-AU <> COD-AU1$.

Articoli scritti da un solo autore.

$$SOLO \leftarrow \Pi_{COD-AR}(ART) \setminus PIU$$

Autori che hanno scritto un articolo da soli

$$AUTORESOLO \leftarrow \Pi_{COD-AU}(SCRIVE \bowtie SOLO)$$

Soluzione

$$\Pi_{Nome,Cognome}(AU \bowtie (\Pi_{COD-AU}(AU) \setminus AUTORESOLO))$$

† **Soluzione Esercizio 12**
Creazione di una vista per il conteggio del numero di citazioni per articolo (con almeno una citazione).

```
CREATE VIEW NCitazioni(COD-AR,NCit) AS
SELECT COD-CITATO, COUNT(*)
FROM   CITAZIONI
GROUP BY COD-CITATO
```

Se si vogliono far comparire anche gli articoli senza citazione (NCit=0) la vista pò essere definita nel seguente modo

```
CREATE VIEW NCitazioni(COD-AR,NCit) AS
SELECT COD-AR, COUNT(COD-RIF)
FROM   ARTICOLI RIGHT OUTER JOIN CITAZIONI ON COD-ART=COD-CITATO
GROUP BY COD-AR
```

Soluzione

```
SELECT R.TITOLO, A.TITOLO
FROM   RIVISTA AS R NATURAL JOIN ARTICOLO AS A NATURAL JOIN NCitazioni
WHERE NCit >= ALL (SELECT NCit
                   FROM   ARTICOLO AS A1 NATURAL JOIN NCitazioni
                   WHERE A1.COD-R = A.COD-R)
```

†
Soluzione Esercizio 13

per semplicità si supponga che tutti campi della descrizione dell'articolo siano di tipo stringa.

```
PROCEDURE BibItem(IN Art : string, OUT bib : string)

VAR
EXEC SQL BEGIN DECLARE SECTION
        titA, titR, NomeA, CognomeA, AnnoA, PagI, PagF : string;
        Num, Cart : string
EXEC SQL END DECLARE SECTION
```

```
BEGIN
Cart := Art; bib:= ”;
EXEC SQL DECLARE scan CURSOR FOR
            SELECT NOME, COGNOME
            FROM   SCRIVE NATURAL JOIN AUTORI
            WHERE COD-AR = :Cart
            ORDER  BY COGNOME, NOME;
EXEC SQL OPEN scan;
EXEC SQL FETCH scan INTO :NomeA, :CognomeA
WHILE SQLCODE = 0 DO
        BEGIN
        bib := bib + :NomeA + ', ' + :CognomeA + '; ';
        EXEC SQL FETCH scan INTO :NomeA, :CognomeA;
        END
EXEC SQL SELECT A.TITOLO, R.TITOLO, A.NUMERO, A.ANNO, A.PAG-I, A.PAG-F
                    INTO :titA, :titR, :Num, :AnnoA, :PagI, :PagF
            FROM   ARTICOLO AS A NATURAL JOIN RIVISTA AS R
            WHERE COD-AR = :Cart
IF SQLCODE = 0 THEN bib := bib + titA + ', ' + titR + ', '
                    + :Num + ', ' + :AnnoA + ', ' + :PagI + ', ' + :PagF + '.'
END

PROCEDURE Bibliografia(IN Art : string, OUT bib : string)

VAR
EXEC SQL BEGIN DECLARE SECTION
            Aus, Cart, Citato : string
EXEC SQL END DECLARE SECTION

BEGIN
Cart := Art; bib:= ”;
EXEC SQL DECLARE scan CURSOR FOR
            SELECT COD-CITATO
            FROM   CITAZIONI
            WHERE COD-RIF = :Cart
EXEC SQL OPEN scan;
EXEC SQL FETCH scan INTO :Citato
WHILE SQLCODE = 0 DO
        BEGIN
        Bibitem(Citato,Aus)
        bib := bib + Aus + ' ';
        EXEC SQL FETCH scan INTO :Citato;
        END
END
```

†

2 Prova scritta del 28-07-04

Si consideri il seguente schema relazionale che descrive la gestione di sale cinematografiche:
$SALA(\underline{CodSala}, Nome, Schermo, Audio)$
$POSTO(CodSala, \underline{CodPosto}, Fila, NumPosto, Fascia)$
$PROIEZIONE(\underline{CodicePr}, Film, CodSala, Data, OraInizio)$
$BIGLIETTO(CodicePr, CodPosto)$.
$SALA$, fornisce alcune caratteristiche tecnice della sala di proiezione;
$POSTO$ descrive i singoli posti a sedere di una sala qualificandoli con il codice della sala (CodSala), il numero della fila (Fila), il numero del posto nella fila (NumPosto), il livello di qualità del posto (Fascia che può assumere i valori 'A', 'B' o 'C', dove 'A' indica la fascia di livello superiore);
$PROIEZIONE$ descrive le proiezioni dei film nelle sale qualificandoli con il film proiettato, il codice della sala, la data e l'ora della proiezione;
$BIGLIETTO$ contiene informazione sui posti venduti (CodPosto) per ogni proiezione (CodicePr).

Esercizio 21 *Si scriva una espressione in algebra relazionale che, se valutata, fornisce il codice delle proiezioni per le quali tutti i posti di fascia 'A' sono stati occupati.*

Esercizio 22 *Scrivere una vista in SQL che per ogni proiezione fornisca la data, il codice della sala, l'ora di proiezione, il numero di posti occupati e il numero di posti liberi (separatamente) per fascia 'A', 'B' e 'C'.*

Esercizio 23 *Si scriva una procedura che riceva in ingresso un codice di proiezione, una fascia di posti (A, B o C) ed il quantitativo N dei posti che si vogliono occupare. La procedura verifica che vi siano almeno N posti della fascia richiesta nella sala della proiezione e in caso favorevole procede all'assegnazione dei primi posti liberi (i posti si suppongono ordinati per Fila e NumPosto). L'assegnazione di un posto viene effettuata inserendo nella tabella BIGLIETTO una riga con la proiezione e il codice del posto assegnato. Se il numero di posti liberi nella fascia richiesta è insufficiente rispetto alla richiesta viene fornita una segnalazione di errore. (Nell'esercizio si assuma di NON poter usare aspetti di SQL che consentono di limitare ad un numero predefinito il numero di righe restituite da una SELECT).*

Esercizio 24 *Nell'ambito della gestione della concorrenza*

1. *Si dica che cosa si intende quando si dice che uno scheduling è serializzabile;*
2. *Si descriva la tecnica di timestamping per la gestione della concorrenza;*
3. *Si assuma che negli scheduling di seguito riportati vi siano le operazioni di tre transazioni distinte (il pedice qualifica la transazione e corrisponde al timestamp della transazione). Si dica negli scheduling considerati quali transazioni vengono eseguite con successo.*
   $S' = R_1(X), R_2(Y), W_1(X), W_2(Y), R_3(Z), R_2(X)$
   $S'' = R_1(X), R_2(Y), R_2(X), W_2(Y), W_1(X), R_3(Z)$
   $S''' = R_2(X), R_3(X), W_3(X), W_2(X), W_1(Z)$

3 Prova scritta del 01-07-04

Si consideri il seguente schema relazionale che descrive il contenuto di libri:
$LIBRI(\underline{ISBN}, TITOLO)$
$AUTORI(ISBN, NOME, COGNOME)$
$CAPITOLI(\underline{ISBN, N-CAP}, PAG-I, PAG-F, TITOLO)$
$FIGURE(\underline{ISBN, N-FIG}, DIDASCALIA, PAG)$.
$LIBRI$ contiene codice ISBN e titolo dei libri; $AUTORI$ contiene la descrizione del nome e del cognome degli autori di un libro; CAPITOLI contiene numero, titolo, pagina iniziale e finale di ogni capitolo di un libro; FIGURE contiene numero, didascalia e pagina di ogni figura di un libro. (Ovviamente un libro può avere più autori, capitoli e figure.)

Esercizio 31 *Senza fare uso delle operazioni di conteggio, si scriva una espressione in algebra relazionale che, se valutata, fornisce i titoli dei libri scritti da un solo autore e senza figure.*

Esercizio 32 *Scrivere una interrogazione in SQL che restituisca il titolo di libri che abbiano al più una figura per capitolo.*

Esercizio 33 *Si supponga che nella base di dati sia stata già creata una tabella di schema $TITOLICAPITOLI(ISBN, CONCTITOLI)$. Per un libro identificato dal suo ISBN, l'attributo CONCTITOLI contine la concatenazione di tutti i titoli dei capitoli (ordinati per numero di capitolo) di quel libro separati da un carattere ';'*
Esempio. Se il libro di ISBN '0-55-1' ha i capitoli di titolo 'Introduzione a SQL', 'Sintassi di SQL' e 'Conclusioni', allora l'attributo CONCTITOLI contiene la stringa 'Introduzione a SQL;Sintassi di SQL;Conclusioni'.
Si scriva una procedura in C o Pascal (senza parametri di ingresso) che ha come effetto la popolazione della tabella $TITOLICAPITOLI$ (inserisce una riga per ogni libro presente nella tabella LIBRI).

Esercizio 34 *Si descriva la struttura di un B^+ albero e si scriva la sequenza di B^+ alberi di grado 3 con chiavi che assumono come valori numeri interi ottenuta, a partire dall'albero vuoto, inserendo la seguente sequenza di valori di chiave: 8, 5, 3, 4, 2, 9, 10.*

Soluzione Esercizio 31
Siano LI, AU, CP, FG le relazione per LIBRI, AUTORI, CAPITOLI, FIGURE, rispettivamente.
ISBN di libri con almeno due autori.

$$PIU \leftarrow \Pi_{ISBN}(\rho_{AU1(ISBN1,NOME1,COGNOME1)}(AU) \bowtie_C AU))$$

dove C è la condizione $ISBN = ISBN1 \wedge (COGNOME <> COGNOME1 \vee NOME <> NOME1)$.

ISBN di libri con figure.

$$FIG \leftarrow \Pi_{ISBN}(FI)$$

Risultato.

$$\Pi_{TITOLO}(LI \bowtie (\Pi_{ISBN}(LI) \setminus (PIU \cup FIG)))$$

† **Soluzione Esercizio 32**

```
SELECT TITOLO
FROM   LIBRI AS
WHERE ISBN NOT IN
        (SELECT C.ISBN
        FROM   (CAPITOLI AS C JOIN FIGURE AS F1 ON C.ISBN=F1.ISBN)
               JOIN FIGURE AS F2 ON C.ISBN=F2.ISBN
        WHERE F1.N-FIG <> F2.N-FIG AND F1.PAG BETWEEN C.PAG-I AND C.PAG-F
               AND F2.PAG BETWEEN C.PAG-I AND C.PAG-F)
```

†

Soluzione Esercizio 33

```
PROCEDURE Scarica

VAR
EXEC SQL BEGIN DECLARE SECTION
            Lib, Cap, StTit, CurrLib, Conc: string;
            NCap : integer
EXEC SQL END DECLARE SECTION

BEGIN
EXEC SQL DECLARE scan CURSOR FOR
            SELECT ISBN, N-CAP, TITOLO
            FROM   CAPITOLI
            ORDER  BY ISBN, N-CAP;
EXEC SQL OPEN scan;
EXEC SQL FETCH scan INTO :Lib, :Cap, :StTit;
IF SQLCODE = 0 THEN
            BEGIN   CurrLib := Lib;
            Conc := StTit
            END
WHILE SQLCODE = 0 DO
        BEGIN
        EXEC SQL FETCH scan INTO :Lib, :Cap, :StTit;
        WHILE SQLCODE = 0 AND CurrLib = Lib DO
                BEGIN
                Conc := Conc + '; ' + StTit;
                EXEC SQL FETCH scan INTO :Lib, :Cap; :StTit;
                END
        EXEC SQL INSERT INTO TITOLICAPITOLI VALUES (:CurrLib, :Conc)
        END
END
```

†

Basi di Dati e Sistemi Informativi I, Prove scritte AA 2005/06

Adriano Peron

Facoltà di Scienze M.F.N., Corso di Laurea in Informatica, Dipartimento di Scienze Fisiche,
Università di Napoli 'Federico II', Italy
E-mail: peron@na.infn.it

1 3 marzo 2007

Si consideri il seguente schema relazionale che descrive una collezione di alberi costruiti su un insieme comune di nodi:
$ALBERI(\underline{CodA}, Radice)$
$NODI(\underline{CodN}, Peso)$
$COMPNODI(CodA, CodN)$
$COMPARCHI(CodA, Padre, Figlio, Peso)$

$ALBERI$ descrive la collezione di alberi presenti ($Radice$ è un codice di nodo);
$NODI(\underline{CodN}, Peso)$, descrive la collezione di nodi condivisi su cui tutti gli alberi sono definiti ($Peso$ è un intero);
$COMPNODI$ descrive l'insieme dei nodi associati ad un albero;
$COMPARCHI$ descrive l'insieme degli archi associati ad un albero (a ciascun arco è associato un peso espresso da un valore intero;

Esercizio 11 (6 punti) *Si scriva una interrogazione in algebra relazionale che fornisca se valutata il codice degli alberi in cui ogni nodo (tranne le foglie) ha esattamente un discendente.*

Esercizio 12 (9 punti) *Si scriva una interrogazione in SQL che fornisca coppie di codici di alberi tali che il secondo elemento della coppia è un sottoalbero del primo elemento della coppia (ogni nodo e ogni arco del sottoalbero deve essere nodo e arco, rispettivamente, del primo albero).*

Esercizio 13 (9 punti) *Si scriva una procedura PLSQL che riceve in ingresso il codice di un albero ed il codice di un nodo e che restituisce la somma dei pesi incontrati negli archi dal nodo dato alla radice e la somma dei pesi incontrati nei nodi dal nodo alla radice.*

Esercizio 14 (9 punti) *Si esprima nel modo più adeguato il seguente insieme di vincoli:*

- *Ogni nodo di un albero ha al più un padre;*
- *La radice di un albero non ha padre;*
- *I nodi padre e figlio di un arco associato ad una albero devono essere nodi dell'albero;*
- *Quando viene rimosso un nodo da un grafo (cancellazione dalla tabella $COMPNODI$) devono essere rimossi tutti gli archi e tutti i nodi del sottoalbero radicato nel nodo (discendenti);*

2 2 febbraio 2007

Si consideri il seguente schema relazionale che descrive alcuni dati anagrafici:
$PERSONA(\underline{CF}, Nome, Cognome, DataN, DataM)$
$RESIDENZA(\underline{CF, DataI}, Via, Num, Citta, DataF)$
$GENITORI(\overline{CF}, \overline{CFMadre}, CFPadre)$

$PERSONA$, fornisce informazioni anagrafiche quali il Codice fiscale (chiave) data di nascita e data di morte (attributo parziale nullo per persone vive); $RESIDENZA$ descrive lo storico delle residenze di ogni persona (la data di fine residenza è attributo parziale ed è in particolare nullo per la residenza corrente); $GENITORI$ associa ad ogni persona padre e madre (entrambi gli attributi posso essere parziali);

Esercizio 21 (6 punti) *Si scriva una interrogazione in algebra relazionale che fornisca se valutata nomi e cognomi delle persone che sono state residenti per tutta la vita nella stessa città.*

Esercizio 22 (9 punti) *Si scriva una interrogazione in SQL che fornisca nome e cognome delle persone che hano avuto la residenza esattamente nello stesso insieme di città del loro padre.*

Esercizio 23 (9 punti) *Si scriva una procedura PLSQL che riceve in ingresso il CF di una persona e restituisce una stringa di caratteri contenete il nome e cognome di tutti gli antenati in linea maschile (padre, nonno paterno, padre del nonno paterno etc.).*

Esercizio 24 (9 punti) *Si esprima nel modo più adeguato il seguente insieme di vincoli:*

- *Ogni persona ha al più un padre ed una madre;*
- *Quando viene cancellata una persona deve essere cancellata anche ogni associazione di parentela che la riguarda;*
- *Non si può essere residenti nello stesso tempo in due luoghi diversi;*
- *Quando viene fissata la data di morte viene anche chiusa in quella data la residenza.*

3 8 gennaio 2007

Si consideri il seguente schema relazionale che descrive la gestione dei piani di studio degli studenti:
$STUDENTE(\underline{Matricola}, Nome, Cognome)$
$PIANI(\underline{CodPiani}, Matricola, Data)$
$COMPOSIZIONE(\underline{CodPiano, CodEsame}, Anno)$
$ESAME(\underline{CodEsame}, Titolo, CFU, Tipo)$
$ANNI(CodEsame, Anno)$
$PROPED(CodEsame, CodEsameProP)$

$PIANI$, fornisce la data di registrazione del Piano di Studi e la matricola dello studente (chiave esterna); $COMPOSIZIONE$ descrive la composizione di un piano di studio (una riga per ogni esame presente nel piano di studi insieme all'anno di corso in cui è previsto); $ESAME$ descrive gli attibuti degli insegnamenti; l'attributo tipo può assumere i valori $OBBLIGATORIO$, $FACOLTATIVO$, $DEFAULT$; $ANNI$ descrive i possibili anni del corso di studi in cui un insegnamento può essere collocato; $PROPED$ descrive i legami di propedeuticità tra i corsi.

Esercizio 31 *(punti 8) Si scriva una espressione in algebra relazionale che, se valutata, fornisca il nome e cognome e la matricola degli studenti che hanno un piano di studi dove tutti gli insegnamenti presenti non hanno richiesta di propedeuticità.*

Esercizio 32 *(punti 9) Definire nel modo più appropriato e semplice il seguente insieme di vincoli: (a) lo stesso esame non deve essere associato più di una volta allo stesso piano di studi; (b) nella tabella $PROPED$ ad un esame possono essere associati solo esami effettivamente presenti nella tabella $ESAMI$; (c) se un esame è presente in un piano di studi devono essere presenti anche tutti gli esami propedeutici; (d) la somma dei CFU presenti in ogni anno (I, II e III) deve essere esattamente 60 CFU.*

Esercizio 33 *(***Punti 9***) Si scriva una procedura in PL\SQL che riceve in ingresso la matricola di uno studente e un nuovo codice di piano di studi (non ancora presente nella base di dati). La procedura crea un piano di studi (avente come codice quello passato per parametro) che contiene tutti gli esami obbligatori e tutti gli esami di default. Se un esame può essere collocato in anni diversi, nel piano di studi viene collocato nel primo anno possibile).*

Esercizio 34 *(***Punti 4***) Si dica se il seguente schedule è compatibile con il locking a due fasi (il pedice funge da identificativo della transazione)*
$R_2(x)W_3(x)c_3W_1(y)c_1R_2(y)W_2(z)c_2$. *Interpretando il pedice come timestamp si dica quali transazioni superano il controllo basato su timestamp.*
Si descriva la differenza tra locking a due fasi e locking a due fasi stretto indicando quali anomalie delle transazioni concorrenti permettono rispettivamente di evitare.

Esercizio 35 *(***Facoltativo. Punti 4***) Si scriva una vista che permette di presentare all'utente in un unico schema tutti i dati presenti in $ESAMI$ ed $ANNI$. Si scriva poi un trigger che permette di gestire (rimpiazzandola) una operazione di INSERT nella vista.*

4 7 settembre 2006

Si consideri il seguente schema relazionale che memorizza alberi con nodi ed archi etichettati:

$TREE(\underline{CodT}, Root)$
$NODI(\underline{CodN}, Label)$
$ARCHI(\underline{CodA}, Padre, Figlio, Label)$
$COMPN(CodT, CodN)$
$COMPA(CodT, CodA)$

$TREE$ fornisce l'elenco degli alberi (codice dell'albero e codice del nodo radice); $NODI$ fornisce l'elenco dei nodi (codice del nodo e dell'etichetta); $ARCHI$ fornisce l'elenco degli archi (codice dell'arco, codice del nodo padre, del nodo figlio ed etichetta); $COMPN$ fornisce l'associazione dei nodi agli alberi; $COMPA$ fornisce l'associazione degli archi agli alberi;

Si osservi che alberi diversi possono condividere nodi ed archi.

Esercizio 41 *Si scriva una interrogazione in algebra relazionale che restituisca il codice degli alberi in cui tutte le foglie siano etichettate dalla etichetta A.*

Esercizio 42 *Si scriva una vista in SQL che per ogni albero fornisca: il numero delle foglie, il numero massimo di figli associato ai nodi, il numero di etichette distinte presenti negli archi.*

Esercizio 43 *Si scriva una funzione in PL\SQL che riceve in ingresso il nome di un albero ed il codice di un nodo appartenente all'albero. La procedura restituice una stringa contenete la concatenazione delle etichette dei nodi e degli archi incontrati nel percorso dal nodo passato in input alla radice.*

Esercizio 44 *Supponendo che le etichette dei nodi siano etichettate da interi, si scriva un Trigger che quando l'etichetta di un nodo viene variata (incrementata o decrementata) propaga la stessa variazione (lo stesso incremento o lo stesso decremento) a tutti i nodi figli.*

5 25 luglio 2006

Si consideri il seguente schema relazionale che descrive una base di dati per la gestione di linee ferroviarie:
$TRATTA(\underline{CodTratta}, StP, StA, Km)$
$TRENO(\underline{CodTreno}, Tipo, StP, StA)$
$PERCORRENZA(\underline{CodTreno, CodTratta}, Stop, OraP, OraA)$

$TRATTA$ contiene la descrizione delle tratte che collegano una stazione di partenza StP, una stazione di arrivo StA e la distanza tra le due stazioni Km. $TRENO$ contiene la descrizione dei treni che percorrono un insieme di tratte da una stazione di partenza ad una di arrivo. La sequenza delle tratte percorse da un treno viene descritta in $PERCORRENZA$: in particolare l'attributo booleano $Stop$ indica se il terno ferma nella stazione di arrivo (se vale $False$ il treno passa senza fermarsi nella stazione di arrivo della tratta); viene riportata inoltre l'ora di partenza e di arrivo relativa alla percorrenza della tratta.

Esercizio 51 *(***Punti 8***) Scrivere una espressione in algebra relazionale che se valutata fornisce il codice dei treni che percorrono percorsi con sole stazioni locali. Una stazione è locale se è destinazione di una ed una sola tratta e stazione di partenza per una e una sola tratta. Inoltre, si scriva una interrogazione che per i treni dell'interrogazione precedente fornisca la lunghezza complessiva del percorso.*

Esercizio 52 *(***Punti 8***) Si scriva una interrogazione in SQL che restituisca coppie di codici di treni che partono dalla stessa stazione, arrivano alla stessa stazione ma fanno un percorso diverso (sequenza di tratte diverse).*

Esercizio 53 *(***Punti 10***) Si scriva una procedura in PL\SQL (o alternativamente in Pascal o C) che riceve in ingresso una stazione e restituisce in uscita una stringa di caratteri che indica tutte le stazioni raggiungibili (connesse da una sequenza di tratte) e la lunghezza complessiva del percorso necessaria per raggiungerle. Per poter ottenere il risultato si faccia uso di una tabella temporanea $TEMP(StR, Km)$ dove ospitare i risultati parziali della procedura. Si assuma che la tabella sia già definita e che all'inizio della procedura vada solo svuotata. Si eviti di inserire duplicati nella tabella (righe per la stessa stazione). Si trascuri il fatto che una stazione potrebbe essere raggiungibile con percorsi diversi e lunghezze di percorso diverse.*

Esercizio 54 *(***Punti 6***) Si scriva uno schema Entità Associazioni per lo schema logico proposto. Si definiscano usando SQL le tre tabelle corrispondenti allo schema logico indicando obbligatoriamente almeno i vincoli di integrità referenziale. A discrezione si suggeriscano altri vincoli ragionevoli per il problema in esame e li si codifichi opportunamente. (il punteggio previsto può essere incrementato in presenza di una adeguata implementazione di vincoli aggiuntivi).*

6 23 giugno 2006

Si consideri il seguente schema relazionale che descrive una base di dati per la programmazione di film e passaggi di spot pubblicitari in un cinema multisala:
$FILM(\underline{CodFilm}, Titolo, Regista, Anno, Durata)$
$PROIEZF(\underline{CodProiez}, CodSala, CodFilm, Data, Ora)$
$SPOT(\underline{CodSpot}, Titolo, Prodotto, Durata)$
$PROIEZS(\underline{CodSpot, CodProiez})$.

$FILM$ contiene la descrizione dei Film da proiettare. $PROIEZ$ contiene la descrizione delle proiezioni previste per un film (il codice del film da proiettare, la sala in cui viene proiettato, la data della proiezione e l'ora di inizio). $SPOT$ contiene la descrizione degli spot pubblicitari da associare alle proiezioni. $PROIEZS$ collega gli spot alle proiezioni (più spot possono essere associati alla medesima proiezione).

Esercizio 61 *(***Punti 8***) Scrivere una espressione in algebra relazionale che se valutata fornisce il titolo dei film nei quali (nel corso delle varie proiezioni del film) sono passati tutti gli spot descritti nella tabella $SPOT$.*

Esercizio 62 *(***Punti 7***) Definire nel modo più appropriato il seguente insieme di vicoli: (a) lo stesso spot non deve essere associato più di una volta alla medesima proiezione; (b) gli spot possono essere associati solo a proiezioni effettivamente presenti nella tabella $PROIEZF$ (c) nella tabella sono presenti solo le proiezioni relative alla data corrente o a date successive alla data corrente (d) per ogni proiezione, la durata complessiva di tutti gli spot ad essa associati deve essere inferiore o uguale ad una costante fissata K.*

Esercizio 63 *(***Punti 10***) Si scriva una procedura in PL\SQL che riceve in ingresso Il codice di un file, il codice di uno spot, il numero di passaggi N previsto per lo spot, una data iniziale per i passaggi dello spot. La procedura associa (inserendo le righe necessarie nella tabella $PROIEZF$, se possibile) lo spot indicato a N proiezioni del film indicato già programmate (presenti nella tabella $PROIEZF$) seguendo i seguenti criteri (1) lo spot non deve essere già associato a quella proiezione, (2) l'inserimento dello spot non deve produrre la violazione del vincolo (d) dell'esercizio precedente, (3) sono preferibili le proiezioni che hanno la data fornita dal parametro o quelle immediatamente successive (non vanno considerate quelle antecedenti).*

Esercizio 64 *(***Punti 5***) Si dica se il seguente schedule è compatibile con il locking a due fasi (il pedice funge da identificativo della transazione)*
$R_2(x)W_3(x)c_3W_1(y)c_1R_2(y)W_2(z)c_2$. *Interpretando il pedice come timestamp si dica quali transazioni superano il controllo basato su timestamp.*
Si descriva la differenza tra locking a due fasi e locking a due fasi stretto indicando quali anomalie delle transazioni concorrenti permettono rispettivamente di evitare.

Esercizio 65 *(***Facoltativo. Punti 5***) Si scriva una vista che permette di presentare all'utente in un unico schema i dati riguardanti il film (almeno il titolo), delle proiezioni e degli spot (almeno il titolo dello spot) associati alle proiezioni. Si scriva poi un trigger che permette di gestire (rimpiazzandola) una operazione di INSERT nella vista.*

Basi di Dati e Sistemi Informativi I, Prove scritte AA 2006/07

Adriano Peron

Facoltà di Scienze M.F.N., Corso di Laurea in Informatica, Dipartimento di Scienze Fisiche,
Università di Napoli 'Federico II', Italy
E-mail: peron@na.infn.it

1 27 marzo 2008

Si consideri il seguente schema relazionale che descrive la strutturazione di un libro.
$LIBRO(\underline{ISBN}, Titolo, Anno)$
$SEZIONI(ISBN, \underline{Codice}, Titolo, Tipo)$
$INCLUSIONE(Contenete, Incluso)$

$LIBRO$ descrive le classi. $SEZIONI$ descrive le sezioni di un libro. La struttura del libro può essere arbitrariamente complessa (una sezione può contenere altre sezioni, le quali possono contenere a loro volta altre sezioni etc) e la struttura che descrive l'inclusione delle sezioni è dunque un albero di altezza arbitraria. Si assume che che esista una sezione che rappresenta l'intero libro (radice nell'albero della struttura). $INCLUSIONE$ indica la relazione di inclusione tra le sezioni.

Esercizio 11 *(8 punti) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce il titolo dei libri che sono costituiti solo da sezioni prive di sottosezioni.*

Esercizio 12 *(8 punti) Si scriva una interrogazione SQL che restituisce il titolo dei libri la cui struttura è un albero di grado k (ogni sezione o non ha sottosezioni oppure ha esattamente k sottosezioni).*

Esercizio 13 *(8 punti) Si scriva un metodo PLSQL che riceve in ingresso il codice di una sezione e che restituisce una stringa di caratteri contenente nell'ordine, il titolo della sezione stessa, la lista di tutti i titoli delle sezioni che la contengono (nell'ordine di risalita dalla sezione alla radice), il titolo del libro.*

Esercizio 14 *(8 punti) Si consideri la relazione* $CORSI(SSD, TITOLO, GRUPPO, SEMESTRE, AA, DOCENTE)$ *che descrive la attivazione di corsi in un determinato Anno Accademico. Valgono le seguenti dipendenze funzionali*

- $Docente \rightarrow SSD;$
- $TITOLO \rightarrow SSD;$
- $TITOLO, AA \rightarrow SEMESTRE;$
- $TITOLO, AA, GRUPPO, SEMESTRE \rightarrow DOCENTE;$

Trovare le chiavi.
Trovare una copertura canonica per le dipendenze.
Dire se $CORSI$ *è in terza forma normale e se è in forma normale di Boyce-Code.*
Se la tabella non è in terza forma normale scomporla applicando l'algoritmo di normalizzazione.

2 8 gennaio 2008

Si consideri il seguente schema relazionale che descrive la strutturazione di classi in un class diagram di UML.
$CLASS(\underline{Nome}, Descrizione)$
$REFINE(SuperClass, SubClass, Total, Disjoint)$
$ASSOCIAZIONI(Nome, Descrizione)$
$COMPASSOC(Associazione, Classe, Ruolo, CardMin, CardMax)$

$CLASS$ descrive le classi. $REFINE$ descrive la relazione di specializzazione delle classi (SubClass è la specializzazione della classe SuperClass); $Total$ e $Disjoint$ sono due tipi booleani che indicano se la specilizzazione è totale e disgiunta. $ASSOCIAZIONI$ indica quali sono le associazioni presenti tra le classi. $COMPASSOC$ indica quali sono le classi che partecipano alla associazione: Associazione è il nome della associazione a cui Classe partecipa; Ruolo indica il nome del ruolo della partecipazione, CardMin e CardMax indicano la crdinalità minima e massima della partecipazione della classe alla associazione.

Esercizio 21 *(7 punti) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce il nome delle classi che NON sono coinvolte in nessuna associazione ricorsiva (è ricorsiva quando la classe partecipa con due ruoli distinti alla associazione).*

Esercizio 22 *(7 punti) Si scriva una vista che per ogni classe indicata dal nome restituisce il numero delle classi che la specializzano, e il numero delle associazioni binarie uno a uno, uno a molti, e molti a molti a cui la classe partecipa.*

Esercizio 23 *(8 punti) Si scriva un metodo PLSQL che riceve in ingresso il nome di una classe e che restituisce una stringa di caratteri contenente, il nome della classe e la lista delle associazioni attributi (nome associazioni e ruolo separati da una virgola) visibili in quella classe (sono visibili nella classe tutte le associazioni della classe e quelle delle sue superclassi).*

Esercizio 24 *(7 punti) Si consideri la relazione* $ORGANICO(Qualifica, Specializ, Anzianita, Stipendio, Reparto)$ *che descrive lo schema di retribuzione delle figure professionali di una azienda. La relazione gode delle seguenti dipendenze logiche:*

- $Specializzazione \rightarrow Reparto;$
- $Qualifica, Specializzazione, Anzianita, Reparto \rightarrow Stipendio;$

Trovare le chiavi.
Trovare una copertura canonica per le dipendenze.
Dire se $ORGANICO$ *è in terza forma normale e se è in forma normale di Boyce-Code.*
Se la tabella non è in terza forma normale scomporla applicando l'algoritmo di normalizzazione.

3 6 settembre 2007

Si consideri il seguente schema relazionale che descrive la strutturazione di classi in un class diagram di UML.
$CLASS(\underline{Nome}, Descrizione)$
$REFINE(SuperClass, SubClass)$
$METODI(\underline{Class, Nome}, TipoOut)$
$PARAMETRI(\underline{Class, NomeM, NomePar}, TipoPar)$

$CLASS$ descrive le classi. $REFINE$ descrive la relazione di specializzazione delle classi (SubClass è la specializzazione della classe SuperClass; una classe non è specializzata se non compare come SuperClass in nessuna riga in REFINE). $METODI$ indica quali metodi vengono associati ad una classe: $Class$ indica il nome della classe di appartenenza, $Nome$ il nome del metodo, $TipoOut$ il tipo di dato restituito dal metodo. $PARAMETRI$ indica quali parametri vengono associati ad un metodo ($Class$ indica il nome della classe di appartenenza del metodo, $NomeM$ il nome del metodo, $NomePar$ il nome del parametro e $TipoPar$ il tipo del parametro).

Esercizio 31 *(7 punti) Si scriva una interrogazione in algebra relazionale (senza l'uso di operazioni di conteggio) che, se valutata, fornisce il nome delle classi non specializzate che hanno esclusivamente metodi con un (ed uno solo) parametro.*

Esercizio 32 *(7 punti) Si scriva una interrogazione SQL che recupera il nome del metodo e della sua classe di appartenenza per metodi che costituiscono un overriding di metodi presenti nella classe padre. Si ricorda che un metodo A costituisce un overrinding del metodo B se A ha lo stesso nome di B ma almeno un parametro diverso da B.*

Esercizio 33 *(9 punti) Si scriva un metodo PLSQL che riceve in ingresso il nome di una classe e che restituisce una stringa di caratteri contenente, il nome della classe e la lista dei nomi dei metodi visibili in quella classe (sono visibili nella classe tutti i nomi associati alla classe e quelli associati alle sue superclassi).*

Esercizio 34 *(7 punti) Si consideri la relazione*
$R(A, B, C, D)$ *e le seguenti dipendenze logiche:*

- $AB \rightarrow C$;
- $A \rightarrow D$;
- $BD \rightarrow C$

Trovare le chiavi.
Trovare una copertura canonica per le dipendenze.
Dire se R è in terza forma normale e se è in forma normale di Boyce-Code.
Se lo schema non è in terza forma normale scomporlo applicando l'algoritmo di normalizzazione.

4 24 luglio 2007

Si consideri il seguente schema relazionale che descrive la strutturazione di classi in un class diagram di UML.
$CLASS(\underline{Nome}, Descrizione)$
$REFINE(SuperClass, SubClass, Total, Disjoint)$
$ATTRIBUTI(Class, Nome, Tipo)$
$BASICTYPE(Nome)$

$CLASS$ descrive le classi. $REFINE$ descrive la relazione di specializzazione delle classi (SubClass è la specializzazione della classe SuperClass); $Total$ e $Disjoint$ sono due tipi booleani che indicano se la specilizzazione è totale e disgiunta. $ATTRIBUTI$ indica quali attributi vengono associati ad una classe. Il tipo di un attributo può essere di tipo basico (contenuto nello schema $BASICTYPE$) oppure il nome di una classe.

Esercizio 41 *(7 punti) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce il nome delle classi non specializzate che prevedono solo attributi di tipo BASICTYPE.*

Esercizio 42 *(7 punti) Si scriva una interrogazione SQL che recupera coppie di nomi di classi in modo che per ogni attributo della seconda classe vi sia un attributo della prima classe che abbia lo stesso nome e lo stesso tipo.*

Esercizio 43 *(9 punti) Si scriva un metodo PLSQL che riceve in ingresso il nome di una classe e che restituisce una stringa di caratteri contenente, il nome della classe e la lista degli attributi (nome e tipo degli attributi separati da una virgola) visibili in quella classe (sono visibili nella classe tutti gli attributi associati alla classe ed quelli associati alle sue superclassi).*

Esercizio 44 *(Facoltativo 5 punti) Si esprimano i seguenti vincoli:*

- *Tutte le indicazioni di specializzazione della stessa classe devono avere gli stessi valori per gli attributi total e disjoint.*
- *I valori dellattributo tipo nello schema $ATTRIBUTI$ devono essere o di tipo $BASICTYPE$ o devono essere nomi di classi.*

Esercizio 45 *(7 punti) Si consideri la relazione* $ORGANICO(Qualifica, Specializ, Anzianita, Stipendio, Reparto)$ *che descrive lo schema di retribuzione delle figure professionali di una azienda. La relazione gode delle seguenti dipendenze logiche:*

- $Specializzazione \rightarrow Reparto;$
- $Qualifica, Specializzazione, Anzianita, Reparto \rightarrow Stipendio;$

Trovare le chiavi.
Trovare una copertura canonica per le dipendenze.
Dire se $ORGANICO$ è in terza forma normale e se è in forma normale di Boyce-Code.
Se la tabella non è in terza forma normale scomporla applicando l'algoritmo di normalizzazione.

5 29 giugno 2007

Si consideri il seguente schema relazionale che descrive la composizione di pezzi meccanici
$PEZZI(\underline{CodP}, Articolo, Modello, Reparto, Scaffale, Quantita, Costo)$
$ASSEMBLAGGIO(Composto, Componente, Quantita)$

$PEZZI$ descrive i pezzi meccanici presenti in magazzino (viene indicato il reparto dove viene custodito il pezzo e lo scaffale nel reparto; Quantità indica il numero di pezzi presenti in magazzino);
$ASSEMBLAGGIO$, descrive il modo in cui i pezzi posso essere assemblati, in particolare *Composto* indica il coddice di un pezzo meccanico che richiede l'assemblaggio interno di altri pezzi indicati dall'attributo *Componente* (è il codice di un pezzo);

Esercizio 51 *(7 punti) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce il codice dei pezzi che sono composti (al primo livello di composizione) da esattamente due (due e solo due) pezzi componenti distinti (Si intenda da due tipologie differenti di pezzi componenti, non da due esemplari dello stesso pezzo). Si risolva l'esercizio preferibilmente senza ricorre ad operazioni di conteggio.*

Esercizio 52 *(7 punti) Un pezzo è di base quando non prevede l'assemblaggio interno di nessun altro pezzo. Si scriva una interrogazione SQL che recupera il codice dei pezzi in cui vengono assemblati SOLO pezzi di base.*

Esercizio 53 *(10 punti) Il costo di un pezzo si ottiene sommando al costo del pezzo stesso, il costo di tutte le sue componenti (si faccia attenzione al fatto che una componente può essere a sua volta composta). Per semplicità si assuma che un pezzo possa occorrere una sola volta nella composizione di un altro pezzo). Si scriva un metodo PLSQL che riceve in ingresso il codice di un pezzo e che restituisce il costo del pezzo. Per il calcolo del costo si suggerisce di usare una tabella temporanea $TEMP(CodP, Costo)$ per memorizzare i risultati intermedi della scomposizione del pezzo nelle sue componenti).*

Esercizio 54 *(7 punti) Si consideri la relazione $MAGAZZINO(Articolo, Modello, Scaffale, Stanza)$ che descrive la collocazione dei modelli degli articoli (un articolo può avere vari modelli) negli scaffali delle stanze di un magazzino (una stanza ha vari scaffali). La relazione gode delle seguenti proprietà:*

- *Gli esemplari di un articolo che hanno lo stesso modello stanno nello stesso scaffale;*
- *Gli esemplari del medesimo articolo stanno nella stessa stanza;*
- *Gli esemplari di un articolo che stanno nella stessa stanza e nello stesso scaffale appartengono allo stesso modello.*

Scrivere le dipendenze suggerite nella descrizione sopra riportata.
Trovare una copertura canonica per le dipendenze.
Dire se la tabella $MAGAZZINO$ è in terza forma normale e se è in forma normale di Boyce-Code.
Se la tabella non è in terza forma normale scomporla applicando l'algoritmo di normalizzazione.

6 I Prova intercorso

Si consideri il seguente schema relazionale che descrive la gestione di un magazzino
$MAGAZZINO(\underline{CodA}, Descrizione, Collocazione, Quantita)$
$ORDINE(\underline{CodO}, DataOrdine, CodFornitore, DataArrivo)$
$COMPORDINE(CodA, CodO, Quantita)$
$FORNITORE(\underline{PI}, RagSociale, Via, Citta)$
$CATALOGO(CodA, PI, Prezzo)$

$MAGAZZINO$ descrive gli articoli presenti in magazzino (Quantità indica il numero di pezzi presenti in magazzino);
$ORDINE$, descrive un ordine di articoli ad un fornitore. La data di arrivo è un attributo parziale il cui valore è inserito solo quando la merce viene fornita dal fornitore;
$COMPORDINE$ descrive la composizione dell'ordine (articoli ordinati e relativa quantità);
$FORNITORE$ descrive l'elenco dei fornitori;
$CATALOGO$ descrive il listino prezzi attuato dai fornitori per gli articoli.

Esercizio 61 *Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce il codice degli ordini e la ragione sociale dei fornitori per ordini non ancora arrivati ed urgenti (un ordine è urgente quando tutti gli articoli dell'ordine hanno scorta zero in magazzino).*

Esercizio 62 *Si scriva una interrogazione in SQL che fornisca le PI dei fornitori che hanno nel catalogo tutti gli articoli che nel magazzino hanno scorta zero.*

Esercizio 63 *Si esprima nel modo più adeguato il seguente insieme di vincoli:*

- *Un articolo non compare più di una volta nello stesso ordine;*
- *Il fornitore di un ordine deve avere in catalogo tutti gli articoli dell'ordine;*
- *La data di arrivo di un ordine è successiva a quella di invio (DataOrdine);*
- *Un ordine può riferirsi solo ad articoli presenti nel magazzino*
- *Quando viene inserita la data di arrivo dell'ordine viene aggiornata automaticamente la quantità degli articoli in magazzino;*

7 II Prova intercorso

Si consideri il seguente schema relazionale che descrive la composizione di pezzi meccanici
$PEZZI(\underline{CodP}, Articolo, Modello, Reparto, Scaffale, Quantita, Costo)$
$ASSEMBLAGGIO(Composto, Componente, Quantita)$

$PEZZI$ descrive i pezzi meccanici presenti in magazzino (viene indicato il reparto dove viene custodito il pezzo e lo scaffale nel reparto; Quantità indica il numero di pezzi presenti in magazzino);
$ASSEMBLAGGIO$, descrive il modo in cui i pezzi posso essere assemblati, in particolare *Composto* indica il coddice di un pezzo meccanico che richiede l'assemblaggio interno di altri pezzi indicati dall'attributo *Componente* (è il codice di un pezzo);

Esercizio 71 *Il costo di un pezzo si ottiene sommando al costo del pezzo stesso, il costo di tutte le sue componenti (si faccia attenzione al fatto che una componente può essere a sua volta composta). Per semplicità si assuma che un pezzo possa occorrere una sola volta nella composizione di un altro pezzo). Si scriva un metodo PLSQL che riceve in ingresso il codice di un pezzo e che restituisce il costo del pezzo. Per il calcolo del costo si suggerisce di usare una tabella temporanea $TEMP(CodP, Costo)$ per memorizzare i risultati intermedi della scomposizione del pezzo nelle sue componenti).*

Esercizio 72 *Si consideri la relazione $MAGAZZINO(Articolo, Modello, Scaffale, Stanza)$ che descrive la collocazione dei modelli degli articoli (un articolo può avere vari modelli) negli scaffali delle stanze di un magazzino (una stanza ha vari scaffali). La relazione gode delle seguenti proprietà:*

- *Gli esemplari di un articolo che hanno lo stesso modello stanno nello stesso scaffale;*
- *Gli esemplari del medesimo articolo stanno nella stessa stanza;*
- *Gli esemplari di un articolo che stanno nella stessa stanza e nello stesso scaffale appartengono allo stesso modello.*

Scrivere le dipendenze suggerite nella descrizione sopra riportata.
Trovare una copertura canonica per le dipendenze.
Dire se la tabella $MAGAZZINO$ è in terza forma normale e se è in forma normale di Boyce-Code.
Se la tabella non è in terza forma normale scomporla applicando l'algoritmo di normalizzazione.

Basi di Dati e Sistemi Informativi I, Prove scritte AA 2007/08

Adriano Peron

Facoltà di Scienze M.F.N., Corso di Laurea in Informatica, Dipartimento di Scienze Fisiche,
Università di Napoli 'Federico II', Italy
E-mail: peron@na.infn.it

1 5 maggio 2008 - Prova intercorso

Si consideri il seguente schema relazionale che descrive la gestione di sale cinematografiche:
$SALA(\underline{CodSala}, Nome, Schermo, Audio, Capienza)$
$POSTO(\underline{CodSala, CodPosto}, Fila, NumPosto)$
$PROIEZIONE(\underline{CodicePr}, Film, CodSala, Data, OraInizio, OraFine)$
$BIGLIETTO(CodicePr, CodSala, CodPosto, Prezzo)$.
$SALA$, fornisce alcune caratteristiche tecnice della sala di proiezione;
$POSTO$ descrive i singoli posti a sedere di una sala qualificandoli con il codice della sala (CodSala), il numero della fila (Fila), il numero del posto nella fila (NumPosto);
$PROIEZIONE$ descrive le proiezioni dei film nelle sale qualificandoli con il film proiettato, il codice della sala, la data e l'ora iniziale e finale della proiezione;
$BIGLIETTO$ contiene informazione sui posti venduti per ogni proiezione (CodicePr).

Esercizio 11 *Si scriva una espressione in algebra relazionale (senza usare operazioni di conteggio) che, se valutata, fornisce il codice delle proiezioni per le quali tutti i posti disponibili sono stati occupati.*

Esercizio 12 *Scrivere una vista in SQL che per ogni film fornisca il numero delle proiezioni già effettuate (la data è precedente alla data corrente), il numero complessivo degli spettatori, e il numero medio di spettatori per proiezione.*

Esercizio 13 *Si fornisca l'implementazione più ragionevole per il seguente insieme di vincoli:*

1. *Il numero di posti di una sala è uguale alla capienza dichiarata nella tabella $SALA$;*
2. *Non deve essere emesso un biglietto per una proiezione inesistente o per un posto inesistente;*
3. *Non si può emettere più di un biglietto per una proiezione su di uno stesso posto;*
4. *Le proiezioni di una giornata in una stessa sala non si devono sovrapporre per orario.*

Esercizio 14 *Si supponga di avere una tabella $RICAVI(Film, Incasso)$ dove per ogni film viene indicato l'ammontare dei biglietti venduti per le proiezioni del film. Si scriva un comando SQL che inserisca nella tabella RICAVI una riga per ogni film presente nelle proiezioni che non sia già presente nella tabella stessa inizializzando l'incasso a 0; Si scriva poi un comando SQL che aggiorna la tabella aggiungendo all'ammontare indicato nei vari film la somma ricavata nelle proiezioni dei film della giornata corrente.*

2 16 giugno - compito e II prova intercorso

Per il compito intero si svolgano i primi quattro esercizi.
Per la seconda prova intercorso si svolgano gli ultimi tre esercizi.

Si consideri il seguente schema relazionale che descrive la strutturazione di un documento nelle sue parti (sezioni, sottosezioni, etc) e l'associazione tra parole chiave e le sezioni dove esse sono presenti.

$SEZIONE(\underline{Documento, CodSezione}, Titolo, Testo)$
$STRUTTURA(Documento, SezContenente, SezContenuta, Posizione)$
$PAROLE(Parola)$
$PRESENZE(Parola, Documento, Sezione)$

$SEZIONE$ descrive le sezioni dei documenti (ogni sezione ha un titolo ed un testo. Una sezione può essere composta da sezioni. Tale composizione è descritta in $STRUTTURA$ dove SezContenente è il codice della sezione composta e SezContenuta è il codice della sottosezione e Posizione è una stringa di caratteri i che rappresenta un intero e che indica che SezContenuta è la i-esima sottosezione di SezContenente.
$PAROLE$ fornisce la lista delle parole interessanti (parole chiave) per il recupero delle sezioni. Se una parola chiave è presente in una sezione, è riportata una riga corrispondente in $PRESENZE$ (Sezione in $PRESENZE$ indica il codice di una sezione).

Esercizio 21 *Si scriva una espressione dell'algebra relazionale che se valutata fornisca tutti i documenti in cui sono presenti tutte le parole chiave.*

Esercizio 22 *Si scrivano nel modo più opportuno i seguenti vincoli:*

- *se per una sezione è presente una sottosezione in posizione $i > 1$, allora deve essere presente per la stessa sezione anche una sottosezione in posizione $i-1$ (per semplicità si può assumere per questo esercizio che il tipo di posizione sia intero).*
- *Nello stesso documento, il titolo delle sezioni è unico.*
- *Quando viene inserita una nuova parola chiave nella tabella $PAROLA$ la tabella $PRESENZE$ deve essere aggiornata con la indicazione delle sezioni che contengono la parola.*
- *La cancellazione di una parola chiave da $PAROLE$ deve avere come conseguenza la cancellazione di tutte le indicazioni di presenza di quella parola nelle sezioni.*

Esercizio 23 *Si scriva un metodo PLSQL che riceve in ingresso il nome di un documento e il codice di una sezione e che restituisce una stringa di caratteri con il titolo della sezione ed il titolo di tutte le sue sottosezioni (a tutti i livelli). Per l'ordinamento dei titoli nella stringa di output si consideri il seguente criterio: ad ogni sezione va associata una stringa che rappresenta il cammino nell'albero delle sottosezioni: alla sezione passata per parametro viene associata la stringa vuota; se s è la stringa per una sezione X che ha una sottosezione Y in posizione h allora la stringa per Y è $s.h$. Per determinare i titoli delle sottosezioni si consiglia di usare una tabella $TEMP(documento, codiceesez, stringa)$ che si suppone essere stata già definita.*

Esercizio 24 *Si consideri la relazione*
$R(A, B, C, D)$ *con le seguenti dipendenze funzionali:*

- $A, B \rightarrow C$;

- $A, B, C \rightarrow D$;
- $C \rightarrow A$;

Trovare le chiavi.
Trovare una copertura canonica per le dipendenze.
Dire quali dipendenze di R soddisfano i requisiti della terza forma normale e equali soddisfano i requisiti della forma normale di Boyce-Code.
Se la tabella non è in terza forma normale scomporla applicando l'algoritmo di normalizzazione.

Esercizio 25 *Si scriva un trigger che quando viene inserita una nuova parola chiave nella tabella $PAROLA$ provveda ad aggiornare la tabella $PRESENZE$ con la indicazione delle sezioni che contengono la parola. Si scriva inoltre un trigger che quando viene inserita una nuova sezione provveda ad aggiornare la tabella $PRESENZE$ relativamente a tutte le parole chiave presenti nella nuova sezione.*

3 21 luglio 2008

Si consideri il seguente schema relazionale che descrive parte dei metadati di uno sche relazionale (la definizione delle tabelle e dei loro attibuti, le chiavi primarie ed esterne eventualmente associate alle tabelel).

$TABELLE(\underline{TabId}, Nome, Schema)$
$ATTRIBUTI(\underline{TabId, Nome}, Tipo, Totale)$
$PK(\underline{TabId, Attributo})$
$FK(\underline{TabId, NomeFK}, RefTabId, OnDel, OnUp)$
$FKK(\underline{TabId, NomeFK}, Att, RefAtt)$.

$TABELLE$ descrive i nomi delle tabelle definite in uno schema. $ATTRIBUTI$ descrive l'insieme di attributi di ciascuna tabella, indicando il tipo e la totatilià dell'attributo (Totale è un booleano). PK indica, se esiste una chiave primaria per una tabella, l'insieme degli attributi che formano la chiave primaria (se non vi è chiave primaria per una tabella non vi sono righe corrispondenti in PK). FK indica le eventuali chiavi esterne per una tabella. Per la chiave esterna viene indicato il nome della tabella referenziata e le azioni OnDelete, OnUpdate da intraprendere in caso di violazione del vincolo di integrità referenziale. Le coppie di attributi corrispondenti (attributo della chiave esterna, attributo referenziato nella tabella referenziata) sono invece riportate nella tabella FKK.

Esercizio 31 *Si scriva una espressione dell'algebra relazionale che se valutata fornisca nome della tabella, numero di attributi presenti nella chiave primaria, numero di chiavi esterne per tabelle in cui tutti gli attributi della chiave primaria sono di tipo integer.*

Esercizio 32 *Si scrivano nel modo più opportuno i seguenti vincoli:*

- *Se in FKK vengono messi in corrispondenza due attributi referenziante-referenziato, allora i due attributi devono essere dello stesso tipo e l'attributo referenziato deve far parte della chiave primaria della tabella referenziata.*
- *Quando viene cancelato un attributo da una tabella, devono essere cancellate tutte le chiavi primarie o esterne che coinvolgono in qualche forma l'attributo (anche referenziandolo).*

Esercizio 33 *Si scriva una funzione PLSQL che riceve in ingresso il nome di uno schema e di una tabella e che restituisce una stringa corrispondente alla definizione nello standard SQL della tabella comprensiva delle eventuali chiavi primaria ed esterne.*

Esercizio 34 *Si consideri la relazione*
$R(A, B, C, D, E)$ *con le seguenti dipendenze funzionali:*

- $A \rightarrow B, E$;
- $B \rightarrow A$;
- $A, C \rightarrow D$;
- $B, C \rightarrow D$;

Trovare le chiavi.
Trovare una copertura canonica per le dipendenze.
Dire quali dipendenze di R soddisfano i requisiti della terza forma normale e quali soddisfano i requisiti della forma normale di Boyce-Code.
Se la tabella non è in terza forma normale scomporla applicando l'algoritmo di normalizzazione.

4 17 settembre 2008

Si consideri il seguente schema relazionale che descrive una base di dati per test. La base di dati contiene la descrizione del tipo di test (TIPOTEST) indicando per ogni tipo di domanda (ad esempio, algebra, geometria, logica) quante domande sono presenti (Quantita) di un determinato livello di difficoltà (Livello). Tale composizione si trova nella tabella COMPTEST. La tabella DOMANDA contiene le domande vere e proprie. Le risposte multiple alle domande sono invece memorizzate nella tabella RISPOSTA. (Per ogni risposta multipla prevista per una domanda vi è una riga corrispondente nella tabella RISPOSTA e la risposta corretta ha l'attributo Corretta con valore assegnato TRUE). La tabella ISTANZE contiene istanze del test. Ogni istanza di test ha un numero progressivo univoco (NTest), ha una tipologia di test associata (CodiceT) ed ha associato un insieme di domande presenti nella tabella DOMANDE.

$TIPOTEST(\underline{CodiceT}, Descrizione)$
$COMPTEST(CodiceT, Tipodomanda, Quantita, Livello)$
$DOMANDA(\underline{CodiceD}, Tipodomanda, Livello, Testo)$
$RISPOSTA(CodiceD, Risposta, Corretta)$
$ISTANZE(NTest, CodiceT, CodiceD)$.

Esercizio 41 *Si scriva una espressione dell'algebra relazionale che se valutata fornisca i codici di istanze di test in cui tutte le domande prevedono lo stesso numero di risposte.*

Esercizio 42 *Si scrivano nel modo più opportuno i seguenti vincoli:*

- *Per ogni domanda ci deve essere una ed una sola risposta indicata come corretta.*
- *Ogni codice test presente in ISTANZE deve corrispondere a un codice di test presente e ogni codice domanda deve corrispondere a un domanda presente.*
- *Ogni istanza di test deve rispettare le regole di composizione per il tipo di test di cui è istanza.*

Esercizio 43 *Si scriva una procedura PLSQL che riceve in ingresso il codice di un tipo di test e che ha come effetto la costruzuione di una istanza di test (inserimento della composizione della istanza nella tabella $ISTANZE$). Il numero della istanza di test generato deve essere ottenuto incrementando di 1 il più grande numero di istanza di test già memorizzata. Per semplicità, se la tipologia di test prevede un numero n di domande di un certo tipo e di un certo livello, si associano al test le prime n domande di quel tipo e di quel livello ottenibili interrogando opportunamente la tabella delle domande.*

Esercizio 44 *Alla luce del problema considerato, si consideri la tabella ISTANZE. Per la tabella si introduca un insieme di dipendenze funzionali ragionevoli rispetto alla descrizione del problema. Si indichino le chiavi, si dica se la tabella è in terza forma normale e in forma di Boyce-Codd. Se non è in una delle due forme normali, si applichi l'algoritmo di normalizzazione.*

5 7 gennaio 2009

Si consideri il seguente schema relazionale che descrive le tabelle definite in un database e le associazioni esistenti tra le tabelle.
$TABELLE(\underline{CodTab}, Nome)$
$ATTRIBUTI(CodTab, Nome, Tipo, Totale)$
$CHAVI(CodTab, NomeAttrib)$
$ASSOCIAZIONI(CodAss, Nome, Grado, Creato)$
$COMPASSOC(CodAss, CodTab, Ruolo, Cardinalita)$

$TABELLE$ descrive le tabelle esistenti nel database. $ATTRIBUTI$ descrive gli attributi associati ad ogni tabella. Per ogni attributo viene fornito il nome, il tipo e la indicazione di parzialitè (Totale =0) o totalità (Totale=1). $CHIAVI$ indica quali sono gli attributi che determinano la chiave primaria della tabella. $ASSOCIAZIONI$ indica quali sono le associazioni presenti tra le tabelle (si pensi alle associazioni di un Class Diagram tra le classi o le associazioni tra le Entità in uno schema Entità-Relazioni). Grado indica il grado dell'associazione e Creato è un attributo che indica se l'associazione è stata già codificata nel database (Creato=1) oppure se deve essere ancora creata (Creato=0). Per ogni legame di una associazione verso una tabella è presente nella tabella $LEGASSOCIAZIONI$ una riga che indica la tabella verso cui si stabilisce il legame, il nome del ruolo del legame, e la cardinalità del legame (il grado di partecipazione di una Tabella all'associazione). I valori per la cardinalità sono 1 o M (molti).

Esercizio 51 *(8 punti) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce il nome delle tabelle che NON sono coinvolte in nessuna associazione ricorsiva (è ricorsiva quando la tabella partecipa con almeno due ruoli distinti alla associazione).*

Esercizio 52 *(8 punti) Si scriva una interrogazione in SQL che restituisca le coppie delle tabelle che NON sono legate da alcuna associazione binaria.*

Esercizio 53 *(8 punti) Si scriva un Trigger che viene attivato quando l'attributo Creato di ASSOCIAZIONI viene posto da 0 a 1. L'effetto è quello di creare una tabella per l'associazione, se si tratta di una associazione molti a molti o di grado maggiore a 2. Gli attributi della tabella creata sono gli attributi chiave delle tabelle coinvolte dall'associazione. Per creazione di una tabella si intende l'inserimento della sua descrizione in TABELLE ed ATTRIBUTI. Se si tratta di una associazione 1 a molti si aggiungono gli attributi chiave della tabella che partecipa con 1 agli attributi della tabella che partecipa con molti*

Esercizio 54 *(8 punti) Si consideri la relazione*
$ORGANICO(Qualifica, Specializ, Anzianita, Stipendio, Reparto)$ *che descrive lo schema di retribuzione delle figure professionali di una azienda. La relazione gode delle seguenti dipendenze logiche:*

- $Specializzazione \rightarrow Reparto;$
- $Qualifica, Specializzazione, Anzianita, Reparto \rightarrow Stipendio;$

Trovare le chiavi.
Trovare una copertura canonica per le dipendenze.

Dire se $ORGANICO$ è in terza forma normale e se è in forma normale di Boyce-Code.
Se la tabella non è in terza forma normale scomporla applicando l'algoritmo di normalizzazione.

6 29 gennaio 2009

Si consideri il seguente schema relazionale che descrive la usuale strutturazione logica di folder e file in un file system.

$FOLDER(\underline{CodFolder}, Nome, Visibile, InFolder)$
$ABILITAZIONE(CodUt, CodFolder)$
$FILE(CodFile, Nome, estensione, dimensione, data, InFolder)$
$Utente(CodUt, Nome, Cognome, Log, Passwd)$.

$FOLDER$ descrive le cartelle. Una catella può essere contenuta in un'altra cartella (att. InFolder che risulta nullo solo se il folder risulta essere la radice della gerarchia). L'attributo $Visibile$ può avere valore 'ALL' (il suo contenuto è visibile a tutti) oppure 'PRIVATE' (il suo contenuto è visibile ai soli utenti abilitati. Gli utenti abilitati ad accedere ai vari folder sono elencati nella tabella $ABILITAZIONE$. Un utente abilitato ad accedere ad un folder è abilitato ad accedere a tutti i file del folder e a tutti i folder visibili contenuti nel folder.

Esercizio 61 *Si scriva una espressione dell'algebra relazionale che se valutata fornisca il nome dei folder PRIVATE che garantiscono comunque l'accesso a tutti gli utenti.*

Esercizio 62 *Si scriva in SQL una interrogazione che restituisca lo stesso risultato dell'esercizio precedente.*

Esercizio 63 *Si scriva un metodo PLSQL che riceve in ingresso il codice di un folder e che restituisce lo spazio di memoria occupato dai file contenuti a tutti i livelli nel folder stesso. Per determinare il valore richiesto si consiglia di usare una tabella temporanea $TEMP$ per memorizzare i risultati intermedi del conteggio (la struttura di TEMP può essere liberamente scelta a seconda delle necessità).*

Esercizio 64 *Si scriva un trigger che vine azionato quando viene revocato ad un utente il permesso di accesso ad un folder (cancellazione della riga in ABILITAZIONE). Il trigger provvede a revocare allo stesso utente le eventuali abilitazioni che esso possiede nei folder contenuti (a tutti i livelli) nel folder in questione. Come nel precedente esercizio si consiglia di usare una tabella temporanea.*

Basi di Dati e Sistemi Informativi I, Prove scritte AA 2003/04

Adriano Peron

Facoltà di Scienze M.F.N., Corso di Laurea in Informatica, Dipartimento di Scienze Fisiche,
Università di Napoli 'Federico II', Italy
E-mail: peron@na.infn.it

1 Scritto del 1 marzo 2010

Si consideri il seguente schema relazionale che descrive un database per la memorizzazione di partite in un gioco per scacchiera.
$PEZZO(\underline{CodiceP}, Tipo, Colore, iX, iY)$
$MOSSA(\underline{Partita, Ordine}, CodiceP, daX, daY, aX, aY)$
$ELIMINA(Partita, Ordine, CodiceP)$

$PEZZO$ formisce la descrizione dei pezzi del gioco (iX e iY indicano le coordinate della posizione iniziale del pezzo sulla scacchiera).
$MOSSA$ memorizza le mosse per ogni partita: Ordine è un intero che indica l'ordine della mossa nella partita; daX e daY sono le coordinate della casella di partaenza e aX, aY sono le coordinate della casella di arrivo. $ELIMINA$ indica i pezzi eliminati durante la partita: ordine indica la mossa che ha portato alla eliminazione.

Esercizio 11 *Si scriva una espressione dell'algebra relazionale che se valutata fornisca tutte le partite in cui sono stati eliminati tutti i pezzi neri.*

Esercizio 12 *Si scriva una vista SQL che risponda al seguente schema logico:* $VISTA(Partita, Orizzontali, V$ *Orizzontali e Verticali indicano il numero delle mosse orizzontali e verticali della partita. Nella vista devono comparire solo le partite che hanno un numero di mosse superiore o uguale alla media delle mosse di tutte le partite.*

Esercizio 13 *Si scriva una procedura PLSQL che riceve in ingresso il codeice di una partita e l'ordine di una mossa. La procedura deve restituire una stringa di terne che descrivono lo stato della scacchiera nella partita al compimento della mossa di ordine fornito. Le terne hanno la forma (X,Y,occ) dove X e Y sono le coordinate della casella e occ vale NULL se la casella è vuota e contiene il codice di un pezzo se, invece, la casella è occupata da un pezzo. (Si osservi che, una casella X,Y contiene un pezzo alla fine della mossa i se il pezzo è posto in quella casella da una mossa j con* $j < i$ *e nessuna altra mossa k con* $j < k < i$ *ha spostato il pezzo o ha eliminato il pezzo mettendoci un altro pezzo al suo posto).*

Esercizio 14 *Si consideri lo schema relazionale*
$R(A, B, C, D)$ *La relazione gode delle seguenti dipendenze logiche:*

- $A, B \rightarrow C$;
- $A, B \rightarrow D$;

- $C \rightarrow A$;
- $D \rightarrow B$.

Dire quali sono le chiavi. Dire quali dipendenze dello schema soddisfano la terza forma normale e quali la forma normale di Boyce-Codd. Ridurre lo schema in forma normale di Boyce-Codd.

Soluzione Esercizio 11
Siano PE, MO, EL, le relazioni per PEZZO, MOSSA, ELIMINA rispettivamente.
Tutti i pezzi neri eliminati in tutte le partite.

$$PP \leftarrow \Pi_{Partita}(MO) \times \sigma_{Colore='Nero'}(PE)$$

Partite in cui non sono stati eliminati tutti i pezzi neri.

$$NO \leftarrow \Pi_{Partita}(PP \setminus \Pi_{Partita,CodiceP}(EL))$$

Soluzione

$$\Pi_{Partita}(MO) \setminus NO$$

†
Soluzione Esercizio 11
La vista viene costruita a partire da viste parziali

```
CREATE VIEW Orizz(Partita,Orizzontali) AS
SELECT Partita, COUNT(*)
FROM   MOSSA M
WHERE M.daY=M.aY
GROUP BY Partita

CREATE VIEW Vertic(Partita,Verticali) AS
SELECT Partita, COUNT(*)
FROM   MOSSA M
WHERE M.daX=M.aX
GROUP BY Partita

CREATE VIEW PartiteC(Partita,NMosse) AS
SELECT Partita, COUNT(*)
FROM   MOSSA M
GROUP BY Partita
```

Soluzione

```
SELECT Partita, Orizzontali, Verticali
FROM   (Orizz NATURAL JOIN Vertic) NATURAL JOIN PartiteC
WHERE NMosse ≥ (SELECT AVG(NMosse)
                FROM PartiteC)
```

†

Soluzione Esercizio 13

Della Funzione viene dato solo uno schema di massima trascurando i dettagli.

Intestazione

CREATE FUNCTION Stato (CPartita : Char(8), NMossa Integer) RETURN VARCHAR2(1000)

La parte rilevante del corpo della procedura verifica la presenza di un pezzo sulla casella. La select viene espressa come unione di due SELECT. La prima verifica che il pezzo nella posizione iniziale che non sia stato spostato o eliminato. La seconda verifica la presenza di pezzi messi da una possa in una casella che non siano stati mossi successivamente o eliminati.

```
FORI IN 1 .. MaxX LOOP
    FORJ IN 1 .. MaxY LOOP
        Trovato := NULL
        SELECTP.CodiceP INTO Trovato
        FROM   PEZZO P
        WHEREP.iX = I AND P.iY = J AND
                NOT EXISTS(SELECT *
                        FROM MOSSSA M
                        WHEREM.Ordine ≤ :NMossa AND
                                M.Partita = :CPartita) AND
                P.CodiceP NOT IN
                        (SELECT E.CodiceP
                        FROM ELIMINA E
                        WHEREE.Ordine ≤ :NMossa AND
                                E.Partita = :CPartita)

        IF trovato IS NULL THEN
        (SELECTM.CodiceP
        FROM   MOSSA M
        WHEREM.partita = :CPartita AND M.aX=I AND M.aY=J AND
                NOT EXISTS(SELECT *
                        FROM MOSSSA S
                        WHERES.Ordine BETWEEN M.Ordine AND :NMossa
                                AND M.Partita = :CPartita
                                AND S.daX = M.aX AND S.daY = M.aY AND
                                M.CodiceP = S.CodiceP ) AND
                M.CodiceP NOT IN
                        (SELECT E.CodiceP
                        FROM ELIMINA E
                        WHEREE.Ordine ≤ :NMossa AND
                                E.Partita = :CPartita))
        IF trovato IS NULL THEN risultato := risultato + '('+I + ',' +J ', NULL)'
        ELSE risultato := risultato +'('+I +',' + J ',' + trovato + ')'
        END IF
        ELSE risultato := risultato + '('+I + ',' + J ','+ trovato + ')'
        END IF
```

Essendo interessati solo alle caselle occupate da un pezzo si usa alternativamente un cursore scritto come segue. Il corpo della procedura semplicemente scandisce il cursore formando la stringa di uscita.

```
CURSOR attuali IS (SELECTP.iX,P.iY,P.CodiceP
                   FROM   PEZZO P
                   WHERENOT EXISTS(SELECT *
                                   FROM MOSSSA M
                                   WHEREM.Ordine ≤ :NMossa AND
                                         M.Partita = :CPartita) AND
                         P.CodiceP NOT IN
                                   (SELECT E.CodiceP
                                   FROM ELIMINA E
                                   WHEREE.Ordine ≤ :NMossa AND
                                         E.Partita = :CPartita)

                   UNION
                   (SELECTM.aX,M.aY,M.CodiceP
                   FROM   MOSSA M
                   WHEREM.partita = :CPartita AND
                         NOT EXISTS(SELECT *
                                   FROM MOSSSA S
                                   WHERES.Ordine BETWEEN M.Ordine AND :NMossa
                                         AND M.Partita = :CPartita
                                         AND S.daX = M.aX AND S.daY = M.aY AND
                                         M.CodiceP = S.CodiceP ) AND
                         M.CodiceP NOT IN
                                   (SELECT E.CodiceP
                                   FROM ELIMINA E
                                   WHEREE.Ordine ≤ :NMossa AND
                                         E.Partita = :CPartita))
```

†

Soluzione Esercizio 14

Le chiavi sono date dai seguenti insiemi $\{A, B\}$, $\{C, D\}$, $\{A, D\}$, $\{B, C\}$.
Tutti gli attibutti sono primi e dunque la relazione rispetta la terza forma normale.
Le dipendenze $C \to A$ e $D \to B$ non rispettano la forma normale di Boyce-Codd.
Scomposizione partendo dalla dipendenza $C \to A$.
$R_1 = \{A, C\}$ con la sola dipendenza $C \to A$.
$R_2 = \{C, B, D\}$ con le seguenti dipendenze $D \to B$, $B, C \to D$, $D, C \to B$.
R_2 non rispetta la forma normale di Boyce-Coddper la dipendenza $D \to B$ e dunque serve un altro passo di scomposizione per la realzione R_2.
R_2 viene scomposta nelle relazioni $R_3 = \{D, B\}$ con dipendenza $D \to B$ e $R_4 = \{D, C\}$ con un insieme vuoto di dipendenze.
Il risultato della scomposizione è dato da R_1, R_3 e R_4 con le dipendenze indicate. †

Basi di Dati e Sistemi Informativi I, Prove scritte AA 2010/11

Adriano Peron

Facoltà di Scienze M.F.N., Corso di Laurea in Informatica, Dipartimento di Scienze Fisiche,
Università di Napoli 'Federico II', Italy
E-mail: peron@na.infn.it

1 Scritto del 2 febbraio 2010

Si consideri il seguente schema relazionale che descrive un database per la gestione di condivisione di file in uno schema peer to peer.
$UTENTE(\underline{UsrId}, MaxUp, MaxDown, Stato)$
$FILE(UsrId, IdFile)$
$SCARICO(IdFile, UsrUp, UsrDown)$
$CODA(IdFile, UsrDown, Time)$

$UTENTE$ formisce il numero massimo di operazioni di Upload e Download simultaneamente possibili per un utente e il suo stato che può assumere lo stato On o Off. $FILE$ indica i file di cui gli utenti sono in possesso. $SCARICO$ indica quali sono le operazioni di scarico in corso con indicazione dell'utente che fa l'upload ed il download. $CODA$ indica le richieste in attesa di scarico, con Time il tempo in cui la richiesta è stata inoltrata.

Esercizio 11 *Si scriva una espressione dell'algebra relazionale che se valutata fornisca lo userid degli utenti che hanno il maggior numero di download in corso.*

Esercizio 12 *(Facoltativo) Si scriva una vista SQL che per ogni utente fornisca il numero di upload e download in corso, il numero di file in attesa di dowload e il tempo massimo di attesa.*

Esercizio 13 *Si scrivano nel modo più opportuno i seguenti vincoli:*

- *nella tabella CODA non ci possono essere più richieste dello stesso utente per lo stesso file.*
- *un utente non può avere un numero di upload in corso superiore al suo MaxUp.*
- *Quando un utente passa da stato On a stato Off tutte le operazioni di scarico che lo coinvolgono sono cancellate.*
- *I file sono associati ad utenti presenti in UTENTE.*

Esercizio 14 *Quando un utente commuta il suo stato da Off a On i file che possiede divengono disponibili. Di conseguenza un trigger cerca di attivare il maggior numero di operazioni di scarico presenti nella tabella CODA usando le disponibilità dell'utente secondo i seguenti criteri: Hanno precedenza le richieste più vecchie; le richieste riguardano file posseduti dall'utente; il numero di operazioni di carico attivate non può superare il MaxUp dell'utente; una richiesta non può essere attivata se l'utente ha già in corso un numero di download pari al numero massimo a lui consentito.*
Attivare una richiesta significa inserire una riga corrispondente nella tabella $SCARICO$ e rimuovere la richiesta dalla tabella $CODA$. Scrivere il trigger in questione.

Esercizio 15 *Si consideri lo schema relazionale*
$R(A, B, C, D)$ *La relazione gode delle seguenti dipendenze logiche:*

- $A, B \rightarrow C$;
- $D \rightarrow C$;
- $A \rightarrow D$;
- $D, C \rightarrow A$.

Dire quali dipendenze dello schema soddisfano la terza forma normale e quali la forma normale di Boyce-Codd. Ridurre lo schema in forma normale di Boyce-Codd.

2 Scritto del 3 marzo 2010

Si consideri il seguente schema relazionale che descrive un database per la memorizzazione di partite in un gioco per scacchiera.
$PEZZO(\underline{CodiceP}, Tipo, Colore, iX, iY)$
$MOSSA(\underline{Partita, Ordine}, daX, daY, aX, aY)$
$ELIMINA(Partita, Ordine, CodiceP)$

$PEZZO$ formisce la descrizione dei pezzi del gioco (iX e iY indicano le coordinate della posizione iniziale del pezzo sulla scacchiera).
$MOSSA$ memorizza le mosse per ogni partita: Ordine è un intero che indica l'ordine della mossa nella partita; daX e daY sono le coordinate della casella di partaenza e aX, aY sono le coordinate della casella di arrivo. $ELIMINA$ indica i pezzi eliminati durante la partita: ordine indica la mossa che ha portato alla eliminazione.

Esercizio 21 *Si scriva una espressione dell'algebra relazionale che se valutata fornisca tutte le partite in cui sono stati eliminati tutti i pezzi neri.*

Esercizio 22 *Si scriva una vista SQL che risponda al seguente schema logico:* $VISTA(Partita, Orizzontali, V$ *Orizzontali e Vericali indicano il numero delle mosse orizzontali e verticali della partita. Nella vista devono comparire solo le partite che hanno un numero di mosse superiore o uguale alla media delle mosse di tutte le partite.*

Esercizio 23 *Si scriva una procedura PLSQL che riceve in ingresso il codeice di una partita e l'ordine di una mossa. La procedura deve restituire una stringa di terne che descrivono lo stato della scacchiera nella partita al compimento della mossa di ordine fornito. Le terne hanno la forma (X,Y,occ) dove X e Y sono le coordinate della casella e occ vale NULL se la casella è vuota e contiene il codice di un pezzo se, invece, la casella è occupata da un pezzo. (Si osservi che, una casella X,Y contiene un pezzo alla fine della mossa i se il pezzo è posto in quella casella da una mossa j con* $j < i$ *e nessuna altra mossa k con* $j < k < i$ *ha spostato il pezzo o ha eliminato il pezzo mettendoci un altro pezzo al suo posto).*

Esercizio 24 *Si consideri lo schema relazionale*
$R(A, B, C, D)$ *La relazione gode delle seguenti dipendenze logiche:*

- $A, B \rightarrow C$;
- $A, B \rightarrow D$;
- $C \rightarrow A$;
- $D \rightarrow B$.

Dire quali sono le chiavi. Dire quali dipendenze dello schema soddisfano la terza forma normale e quali la forma normale di Boyce-Codd. Ridurre lo schema in forma normale di Boyce-Codd.

3 Scritto del 9 aprile 2010

Si consideri il seguente schema relazionale che descrive un frammento dei metadati di Oracle 10g.
$UserTables(TableName)$
$UserTabColumns(TableName, ColumnName, DataType, Position)$
$UserConstraints(TabelName, ConstraintName, ConstraintType, SearchCondition, RConstraintName)$
$UserConsColumns(ConstraintName, TableName, ColumnName, Position)$

$UserTabColumns$ descrive gli attributi associati alle colonne (nome colonna e tipo di dato e posizione). $UserConstraints$ descrive i vincoli associati ad ogni tabella. In particolare, ConstraintType assume il valore C per vincolo Check (l'attributo SearchCondition contiene il vincolo associato al Check), P per vincolo PRIMARY KEY, R per vincolo FOREIGN KEY, U per vincolo UNIQUE. Un vicolo FOREIGN KEY in una tabella A fa riferimento ad un corrispondente vincolo PRIMARY KEY della tabella referenziata B mediante l'attributo RConstraintName (indica il nome del vincolo di chiave primaria nella tabella B a cui la chiave esterna della tabella A fa riferimento). Si assuma al fine dell'esercizio che il nome del vincolo sia chiave. $UserConsColumns$ indica il nome degli attributi (e la loro posizione nel vincolo) che formano vincoli di tipo P,R e U.

Esercizio 31 *Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce il nome delle tabelle che hanno il massimo numero di attributi di tipi INTEGER nella chiave primaria.*

Esercizio 32 *Si scriva una interrogazione in SQL che restituisca le coppie dei nomi delle tabelle che sarebbero compatibili per operazioni insiemistiche (tutti gli attibuti in uguale posizione devono avere lo stesso tipo).*

Esercizio 33 *Si scriva una funzione che prende in ingresso il nome di una tabella e restituisce una stringa contenete la dichiarazione secondo la sintassi di SQL dei vincoli PRIMARY KEY ed UNIQUE presenti nella tabella (si ricordi che in una tabella può esserci più di un vincolo UNIQUE).*

Esercizio 34 *Si consideri la relazione*
$R(A, B, C, D)$. *La relazione gode delle seguenti dipendenze logiche:*

- $D \rightarrow A$;
- $C, B \rightarrow D$;
- $A \rightarrow B$.

Trovare tutte le chiavi. Dire se lo schema soddisfa la forma normale di Boyce Codd e in caso negativo applicare l'algoritmo di scomposizione (Boyce Codd) fino ad ottenere la forma normale. Dire se la forma normale ottenuta preserva i dati e le dipendenze della forma di partenza (si dica preliminarmente che cosa significa preservare dati e dipendenze).

4 Scritto del 22 giugno 2010

Si consideri il seguente schema relazionale che descrive un data base per articoli scientifici su rivista.

$RIVISTA(\underline{ISNN}, Nome, Editore)$
$ARTICOLO(\underline{CodA}, ISNN, Titolo, Volume, Numero, Anno, pagI, pagF)$
$AUTORE(\underline{CodAT}, Cognome, Nome, Istituto)$
$SCRIVE(CodA, CodAT, posizione)$
$REFERENZA(CodA, CodARiferito)$

$RIVISTA$ descrive gli attributi della rivista. $ARTICOLO$ raccoglie istanze di articolo. Un articolo è associato ad una rivista tramite la chiave esterna ISNN. $AUTORE$ raccoglie le istanze degli autori di articoli. L'associazione tra articoli e autori viene stabilita dallo schema $SCRIVE$, dove CodA è chiave esterna verso un articolo, CodAT verso autore, e *posizione* è un intero che indica l'ordine dell'autore nella lista degli autori di un articolo (primo autore, secondo autore...etc). Lo schema $REFERENZA$ indica gli articoli citati da un articolo di codice $CodA$ (CodA refernzia CodARiferimento).

Esercizio 41 *Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce il titolo degli articoli in cui gli autori compaiono in ordine alfabetico.*

Esercizio 42 *Si implementino in modo opportuno i seguenti vincoli:*

1. *Due articoli pubblicati nello stesso numero dello stesso volume della stessa rivista non possono sovrapporsi nel numero delle pagine.*
2. *Due autori del medesimo articolo non possono stare nella stessa posizione nell'elenco degli autori.*
3. *le referenze fanno riferimento ad articoli esistenti nel database.*
4. *un articolo che cita un altro articolo non può essere pubblicato in un anno precedente a quello dell'articolo citato.*

Esercizio 43 *Si supponga che sia stata gi definita una tabella*
$CITAZIONI(CodA, NumCitPure, NumCitSelf)$
dove CodA è un articolo, NumCitPure è il numero di volte che viene citato tramite una referenza da un articolo che non ha autori in comune con gli autori di CodA e NumCitPure è il numero di volte che viene citato tramite una referenza da un articolo che ha autori in comune con gli autori di CodA. Si scrivano dei trigger che realizzino i seguenti vicoli. Quando viene inserito un articolo viene inserito in CITAZIONI una riga corrispondente con il numero di citazioni inizializzato a 0. Quando viene inserita una riga in REFERENZE viene aggiornato il conteggio delle citazioni in modo opportuno.

Esercizio 44 *Si scriva una funzione PLSQL che dato in ingresso il codice di un autore restituisca in uscita la descrizione (stringa di caratteri) dei suoi articoli (in ordine crescente di anno). Per ciascun articolo si indichino gli autori, il titolo, il nome della rivista, volume, numero, anno e pagine.*

5 Scritto del 16 luglio 2010

Si consideri il seguente schema relazionale che descrive un database per questionari a risposta multipla (test).
$TEST(\underline{CodT}, Data, Utente, Risultato)$
$DOMANDE(\underline{CodD}, domanda, tipo, livello)$
$RISPOSTE(\underline{CodD, ordine}, corretta, risposta)$
$COMPTEST(\underline{CodT, CodD}, Ordine, OrdineRisposta)$.

$TEST$ memorizza collezioni di istanze di questionari somministrati ad utenti. Risultato è attributo parziale (viene associato un valore solo alla correzione del questionario). $DOMANDE$ memorizza una collezione di domande con le quali i test sono composti. Per ogni domanda vi sono più risposte ($RISPOSTE$) e delle risposte una soltant è corretta (attributo corretto con valore 1). Ordine (A,B,C etc ....) indica l'ordine in cui viene presentata la risposta. $COMPTEST$ descrive la effettiva composizione di una istanza di questionario: CodT indica la istanza di questionario, CodD la domanda che vi fa parte, Ordine (1,2,3...) indica l'ordine di presentazione della domanda, OrdineRisposta indica la scelta dell'utente tra le risposte per la domanda.

Esercizio 51 *Si scriva una espressione dell'algebra relazionale che se valutata fornisca i codici dei test in cui vi sia al più una domanda sbagliata tra le domande dello stesso tipo.*

Esercizio 52 *Si immagini che i test vengano prima predisposti nel database lasciando le risposte a NULL e che le risposte vengano successivamente caricate aggiornando l'attributo Risposta. Si scriva un trigger il quale provvede ad aggiornare il campo risultato nella tabella TEST quando vengono caricate le risposte.*
Per il calcolo del punteggio si segua la regola: la risposta non data vale 0, la risposta corretta vale 1, la risposta sbagliata vale - (1 / numerorisposte). Si ragioni su come contenere il numero di volte in cui si fa l'aggiornamento dell'attributo risultato suggerendo quale startegia andrebbe adottata.

Esercizio 53 *Si scriva una funzione che riceve in ingresso il codice di un questionario e fornisce in uscita il testo del questionario. Le domande devono essere nell'ordine previsto. Le risposte a ciascuna domanda devono essere nell'ordine previsto. La risposta scelta dall'utente deve avere come prefisso i caratteri '(X)'.*

Esercizio 54 *Si consideri lo schema relazionale*
$R(A, B, C, D)$ *La relazione gode delle seguenti dipendenze logiche:*

- $A, B \rightarrow C$;
- $D \rightarrow C$;
- $A \rightarrow D$;
- $D, C \rightarrow A$.

Dire quali dipendenze dello schema soddisfano la terza forma normale e quali la forma normale di Boyce-Codd. Ridurre lo schema in forma normale di Boyce-Codd.

6 Scritto del 15 settembre 2010

Si consideri la bozza di Class Diagram che descrive l'anagrafe di una città. Una persona può aver avuto nella sua vita diverse residenze, qualificate da una data di inizio (DATAI) e da una data di fine (DATAF). Per la residenza attualmente in corso la DATAF ha valore nullo. Per persone viventi la data di morte (DATAM) ha valore nullo. Nella tabella $STATOF$ viene indicata la composizione del nucleo familiare: CapoF indica il CF del capofamiglia, membro indica il CF del membro del nucleo famigliare e realzione indica la relazione che lega il Membro a CapoF: coniuge, figlio, padre, etc.

$VIA(\underline{CodV}, Nome)$
$RESIDENZA(\underline{CF, DATAI}, DATAF, CodV, Numero)$
$RESIDENTE(\underline{CF}, Nome, Cognome, DataN, DataM)$
$STATOFAM(CapoF, Membro, relazione)$

Esercizio 61 *(Punti 8) Scrivere una espressione in algebra relazionale che fornisca per ogni via, il numero delle persone viventi che hanno avuto sempre la residenza in quella via (attenzione, potrebbero eventualmente aver cambiato il numero civico di residenza).*

Esercizio 62 *(Punti 8) Si implementino nel modo più opportuno i seguenti vincoli:*

- *Quando viene apposta la data di morte di un residente viene anche contestualmente chiusa con la stessa data la sua residenza.*
- *Tutti i membri dello stesso nucleo familiare devo avere la medesima residenza.*
- *Una persona può avere al più una residenza correntemente attiva.*
- *Una persona può essere a capo di un unico nucleo familiare.*

Esercizio 63 *(Punti 8) Si scriva una procedura PLSQL che riceve in ingresso il codice di una via e un numero civico e che restituisce una stringa contenente la descrizione dei nuclei familiari residentia a quell'indirizzo. L'elenco deve essere ordinato lessicograficamente per cognome-nome del capo famiglia. Per ogni nucleo, va indicato l'elenco (cognome-nome-età) dei membri del nucleo familiare ordinato in modo decrescente per età.*

Esercizio 64 *(Punti 8) Si consideri la relazione*
$R(A, B, C, D)$ *con le seguenti dipendenze funzionali:*

- $A, B \rightarrow C$;
- $A, B, C \rightarrow D$;
- $C \rightarrow A$;

Trovare le chiavi.
Trovare una copertura canonica per le dipendenze.
Dire quali dipendenze di R soddisfano i requisiti della terza forma normale e equali soddisfano i requisiti della forma normale di Boyce-Code.
Se la tabella non è in terza forma normale scomporla applicando l'algoritmo di normalizzazione.

Basi di Dati e Sistemi Informativi I, Prove scritte AA 2010/11

Adriano Peron

Facoltà di Scienze M.F.N., Corso di Laurea in Informatica, Dipartimento di Scienze Fisiche,
Università di Napoli 'Federico II', Italy
E-mail: peron@na.infn.it

1 Scritto del 20 dicembre 2010 - Intercorso A

Si consideri il seguente schema relazionale che descrive operazioni di acquisto on-line con carrello (il cliente mette nel carrello gli articoli scelti; alla fine dell'operazione decide se procedere all'ordine indicando che il carrello è completo; quando il carrello è completo il carrello viene trasformato in ordine)
$CLIENTE(\underline{CodCl}, Nome, Cognome, Punti)$
$ORDINI(\underline{CodO}, CodCl, Data)$
$COMPORD(CodO, CodA, Quantita)$
$CARRELLO(CodO, Data, CodCl, Completo)$
$COMPCARRELLO(CodO, CodA, Quantita)$
$ARTICOLO(\underline{CodA}, Costo, Punti, Scorta)$.
$CLIENTE$, fornisce la descrizione del cliente (Punti fornisce i punti cumulati dal cliente per gli acquisti fatti); $ORDINI$ forisce la descrizione di un ordine completato (CodCl indica il codice cliente). $COMPORD$ indica la composizione dell'ordine (Codice dell'ordine, codice dell'articolo e quantità dell'articolo). $CARRELLO$ descrive un ordine non ancora consolidato (Completo è NULL se il carrello non è stato completato e NOT NULL altrimenti). $COMPCARRELLO$ indica la composizione del carrello (Codice dell'ordine, codice dell'articolo e quantità dell'articolo). $ARTICOLO$ descrive gli articoli in vendita (Codice, Costo unitario, Punti guadagnati per ogni singolo acquisto, scorta in magazzino dell'articolo).

Esercizio 11 *Si scriva una espressione in algebra relazionale (senza usare operazioni di conteggio) che, se valutata, fornisce il codice degli utenti che in tutti i loro ordini hanno comperato sempre almeno due articoli.*

Esercizio 12 *Si scriva una interrogazione SQL (preferibilmente senza uso di viste) che, se valutata, fornisce il codice degli utenti che (complessivamente nei vari ordini) hanno comperato tutti gli articoli disponibili.*

Almeno uno dei due seguenti esercizi.

Esercizio 13 *Si fornisca l'implementazione più ragionevole per i seguenti vincoli:*

1. *La scorta di un articolo non deve essere inferiore alla quantità complessiva dell'articolo presente nella composizione dei carrelli (ci possono essere più carrelli);*
2. *Ogni cliente può avere al più un carrello;*
3. *Gli ordini e i carrelli sono associati solo a clienti noti.*

Esercizio 14 *Si supponga che il carrello di codice 'YYY' venga completato. Il completamento del carrello ha il seguente effetto: il carrello e la sua composizione viene trasformato in ordine (conservando il codice) e rimosso dalla tabella dei carrelli; Le scorte di magazzino vengono aggiornate stornando i quantitativi indicati nel carrello. I punti del cliente vengono aggiornati con i punti acquisiti dall'acquisto).*

1. *Si indichi a parole quali operazioni vengo fatte e in che ordine.*
2. *Si scriva in SQL l'operazione per l'aggiornamento della scorta in magazzino.*

2 Scritto del 20 dicembre 2010 - Intercorso B

Si consideri il seguente schema relazionale che descrive operazioni di acquisto on-line con carrello (il cliente mette nel carrello gli articoli scelti; alla fine dell'operazione decide se procedere all'ordine indicando che il carrello è completo; quando il carrello è completo il carrello viene trasformato in ordine)
$CLIENTE(\underline{CodCl}, Nome, Cognome, Punti)$
$ORDINI(\underline{CodO}, CodCl, Data)$
$COMPORD(CodO, CodA, Quantita)$
$CARRELLO(CodO, Data, CodCl, Completo)$
$COMPCARRELLO(CodO, CodA, Quantita)$
$ARTICOLO(\underline{CodA}, Costo, Punti, Scorta)$.
$CLIENTE$, fornisce la descrizione del cliente (Punti fornisce i punti cumulati dal cliente per gli acquisti fatti); $ORDINI$ forisce la descrizione di un ordine completato (CodCl indica il codice cliente). $COMPORD$ indica la composizione dell'ordine (Codice dell'ordine, codice dell'articolo e quantità dell'articolo). $CARRELLO$ descrive un ordine non ancora consolidato (Completo è NULL se il carrello non è stato completato e NOT NULL altrimenti). $COMPCARRELLO$ indica la composizione del carrello (Codice dell'ordine, codice dell'articolo e quantità dell'articolo). $ARTICOLO$ descrive gli articoli in vendita (Codice, Costo unitario, Punti guadagnati per ogni singolo acquisto, scorta in magazzino dell'articolo).

Esercizio 21 *Si scriva una espressione in algebra relazionale (senza usare operazioni di conteggio) che, se valutata, fornisce il codice degli utenti che (complessivamente nei vari ordini) hanno comperato tutti gli articoli disponibili.*

Esercizio 22 *Si scriva una interrogazione SQL (preferibilmente senza uso di viste) che, se valutata, fornisce il codice degli utenti che in tutti i loro ordini hanno comperato sempre almeno due articoli.*

Almeno uno dei due esercizi.

Esercizio 23 *Si fornisca l'implementazione più ragionevole per i seguenti vincoli:*

1. *I punti di un cliente devono esere inferiori o uguali alla somma di tutti i punti cumulati dal cliente negli ordini del 2010;*
2. *Ogni cliente può fare al più un ordine al giorno;*
3. *Le composizioni di ordini e di carrelli sono associati solo ad articoli trattati in magazzino.*

Esercizio 24 *Si supponga che il carrello di codice 'YYY' venga completato. Il completamento del carrello ha il seguente effetto: il carrello e la sua composizione viene trasformato in ordine (conservando il codice) e rimosso dalla tabella dei carrelli; Le scorte di magazzino vengono aggiornate stornando i quantitativi indicati nel carrello. I punti del cliente vengono aggiornati con i punti acquisiti dall'acquisto).*

1. *Si indichi a parole quali operazioni vengo fatte e in che ordine.*
2. *Si scriva in SQL l'operazione per l'aggiornamento del punteggio del cliente.*

3 Scritto del 2 febbraio 2011

Per il compito intero si svolgano i primi quattro esercizi.
Per la seconda prova intercorso si svolgano gli ultimi tre esercizi.

Si consideri il seguente schema relazionale che descrive la strutturazione di un documento nelle sue parti (sezioni, sottosezioni, etc) e l'associazione tra parole chiave e le sezioni dove esse sono presenti.

$SEZIONE(\underline{Documento, CodSezione}, Titolo, Testo)$
$STRUTTURA(\underline{Documento, SezContenente}, SezContenuta, Posizione)$
$PAROLE(Parola)$
$PRESENZE(Parola, Documento, Sezione, Offset)$

$SEZIONE$ descrive le sezioni dei documenti (ogni sezione ha un titolo ed un testo. Una sezione può essere composta da sezioni. Tale composizione è descritta in $STRUTTURA$ dove SezContenente è il codice della sezione composta e SezContenuta è il codice della sottosezione e Posizione è una stringa di caratteri i che rappresenta un intero e che indica che SezContenuta è la i-esima sottosezione di SezContenente.
$PAROLE$ fornisce la lista delle parole interessanti (parole chiave) per il recupero delle sezioni. Se una parola chiave è presente in una sezione, è riportata una riga corrispondente in $PRESENZE$ (Sezione in $PRESENZE$ indica il codice di una sezione e offset indica la posizione della parola nella sezione; si ricorda che in ORACLE la funzione INSTR(stringa, pattern, [inizio, [occorrenza]]) restituisce, se presente, la posizione del pattern nella stringa)

Esercizio 31 *(7 punti) Si scriva una espressione dell'algebra relazionale che se valutata fornisca tutti i documenti in cui sono presenti tutte le parole chiave.*

Esercizio 32 *(7 punti) Si scriva una interrogazione SQL che che per ogni documento fornisca tutte le coppie di parole chiavi adiacenti nel documento (si trovano nella stessa sezione e non sono separate da altra parola chiave).*

Esercizio 33 *(7 punti) Si scriva una funzione PLSQL che riceve in ingresso il codice di due documenti e che restituisce un intero che rappresenta il numero di parole che i due documenti hanno in comune.*
Si utilizzi la funzione definita per scrivere una vista SIMILITUDINE(Doc1,Doc2,NumParoleComuni) che per ogni coppia di documenti Doc1 e Doc2 riporta il numero di parole in comune.

Esercizio 34 *(6 punti un trigger, 9 due trigger) Si scriva almeno uno dei due trigger seguenti. Si scriva un trigger che quando viene inserita una nuova parola chiave nella tabella $PAROLA$ provveda ad aggiornare la tabella $PRESENZE$ con la indicazione dei documenti, delle sezioni che contengono la parola (e i relativi offset). Si scriva inoltre un trigger che quando viene inserita una nuova sezione provveda ad aggiornare la tabella $PRESENZE$ relativamente a tutte le parole chiave presenti nella nuova sezione.*

Esercizio 35 *(Esercizio facoltativo, 10 punti) Si scriva una procedura PLSQL che riceve in ingresso una stringa di parole chiave del formato @parola1@parola2@... @parolak@. Usando SQL dinamico si scriva una interrogazione che trova i documenti che contengono tutte le parole indicate nella stringa (si ricorda che in SQL la funzione SUBSTR(stringa, pos, lungh) restituisce la porzione della sottostringa di stringa che*

ha inizio in pos ed ha lunghezza lungh). Il risultato della interrogazione deve essere restituito in una stringa.

4 Scritto del 2 febbraio 2011 - bis

Per il compito intero si svolgano i primi quattro esercizi.
Per la seconda prova intercorso si svolgano gli ultimi tre esercizi.

Si consideri il seguente schema relazionale che descrive la strutturazione di un documento nelle sue parti (sezioni, sottosezioni, etc) e l'associazione tra parole chiave e le sezioni dove esse sono presenti.

$SEZIONE(\underline{Documento, CodSezione}, Titolo, Testo)$
$STRUTTURA(\underline{Documento, SezContenente}, SezContenuta, Posizione)$
$PAROLE(Parola)$
$PRESENZE(Parola, Documento, Sezione, Offset)$

$SEZIONE$ descrive le sezioni dei documenti (ogni sezione ha un titolo ed un testo. Una sezione può essere composta da sezioni. Tale composizione è descritta in $STRUTTURA$ dove SezContenente è il codice della sezione composta e SezContenuta è il codice della sottosezione e Posizione è una stringa di caratteri i che rappresenta un intero e che indica che SezContenuta è la i-esima sottosezione di SezContenente.
$PAROLE$ fornisce la lista delle parole interessanti (parole chiave) per il recupero delle sezioni. Se una parola chiave è presente in una sezione, è riportata una riga corrispondente in $PRESENZE$ (Sezione in $PRESENZE$ indica il codice di una sezione e offset indica la posizione della parola nella sezione; si ricorda che in ORACLE la funzione INSTR(stringa, pattern, [inizio, [occorrenza]]) restituisce, se presente, la posizione del pattern nella stringa)

Esercizio 41 *(10 punti) Si scriva una funzione PLSQL che riceve in ingresso il codice di due documenti e che restituisce un intero che rappresenta il numero di parole che i due documenti hanno in comune.*
Si utilizzi la funzione definita per scrivere una vista SIMILITUDINE(Doc1,Doc2,NumParoleComuni) che per ogni coppia di documenti Doc1 e Doc2 riporta il numero di parole in comune.

Esercizio 42 *(10 punti un trigger, 18 due trigger) Si scriva almeno uno dei due trigger seguenti. Si scriva un trigger che quando viene inserita una nuova parola chiave nella tabella $PAROLA$ provveda ad aggiornare la tabella $PRESENZE$ con la indicazione dei documenti, delle sezioni che contengono la parola (e i relativi offset). Si scriva inoltre un trigger che quando viene inserita una nuova sezione provveda ad aggiornare la tabella $PRESENZE$ relativamente a tutte le parole chiave presenti nella nuova sezione.*

Esercizio 43 *(Esercizio facoltativo, 15 punti) Si scriva una procedura PLSQL che riceve in ingresso una stringa di parole chiave del formato @parola1@parola2@... @parolak@. Usando SQL dinamico si scriva una interrogazione che trova i documenti che contengono tutte le parole indicate nella stringa (si ricorda che in SQL la funzione SUBSTR(stringa, pos, lungh) restituisce la porzione della sottostringa di stringa che ha inizio in pos ed ha lunghezza lungh). Il risultato della interrogazione deve essere restituito in una stringa.*

5 Scritto del 2 marzo 2011

Si consideri il seguente schema relazionale che descrive una versione elementare dei dati per un social network
$Utenti(\underline{id}, nome, cognome, mail, passwd)$
$Amici(Utente1, Utente2, Data)$
$Interessi(Utente, Categoria, Valore)$
$Richiesta(Richiedente, Invitato, Tipo, Data, Stato)$
$Post(\underline{codice}, Utente, testo, data)$
$Commento(codicePost, Utente, testo, data)$.

Amici descrive la relazione simmetrica di amicizia tra utenti e la data in cui è stata stabilita; *Interessi* descrive gli interessi di un utente mediante la categoria di interesse (ad es. Sport) e il tipo di interesse (ad es. valore tennis). *Richiesta* descrive le richieste inoltrate dall'utente Richiedente all'utente Invitato, il tipo richiesta (ad. es Amicizia), lo stato della richiesta (attesa,accettata); un elemento di *Post* indica un elemento testuale pubblicato da un utente; un elemento di *Commento* è la risposta di un utente al post individuato da codicePost.

Esercizio 51 *(8 punti) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce il nome degli utenti che hanno scritto post che hanno ricevuto sempre almeno 5 commenti.*

Esercizio 52 *(8 punti) Si implementino in modo opportuno i seguenti vincoli:*

1. *quando una richiesta di amicizia viene accettata la relazione Amici viene aggiornata stabilendo in modo simmetrico la relazione di amicizia.*
2. *Un utente non può commentare più di una volta lo stesso post;*
3. *Lo stato di una richiesta può avere solo o valore Attesa o valore Accettato o valore Rifiutato;*
4. *Non ci può essere una relazione di amicizia tra due utenti senza una richiesta di amicizia accettata.*

Esercizio 53 *(10 punti) Si scriva una funzione che prende in ingresso il codice di un utente ed un numero intero k. La funzione deve recuperare le catene di amici fino al grado k (di grado 1 sono gli amici ufficiali, di grado 2 gli amici degli amici ect.). Ci si avvalga di una tabella temporanea TMP che si suppone già creata. Una volta recuperati gli amici la funzione deve restituirli in una stringa in ordine crescente di grado di amicizia e, per gli amici dello stesso grado, in ordine lessicografico per cognome e nome.*

Esercizio 54 *(10 punti) Usando sql dinamico si scriva una funzione che riceve in ingresso una stringa di lunghezza variabile di caratteri del tipo ?categoria1?valore1?categoria2?valore2?.... e che restituisca in una stringa gli identificativi degli utenti separati da virgole che hanno interessi che soddisfano almeno una delle coppie ?categoria?valore? (si usino le funzioni INSTR e SUBSTR per la scansione della lista di ingresso).*

6 Scritto del 25 marzo 2011

Si consideri il seguente schema relazionale che descrive un database per la gestione di condivisione di file in uno schema peer to peer.
$UTENTE(\underline{UsrId}, MaxUp, MaxDown, Stato)$
$FILE(IdFile, IdUtente)$
$BLOCCHI(IdFile, IdUtente, IdBlocco)$
$SCARICO(IdFile, UsrUp, UsrDown, IdBlocco)$
$RICHIESTA(IdFile, UsrDown, Time, IdBlocco)$

$UTENTE$ formisce il numero massimo di operazioni di Upload e Download simultaneamente possibili per un utente e il suo stato che può assumere lo stato On o Off. $FILE$ indica i file completi di cui gli utenti sono in possesso. Un utente può essere in possesso di un file completo (in questo caso si ha un record in FILE) oppure può essere in possesso solo di alcuni blocchi del file e non di tutti (in questo caso per ogni blocco posseduto si ha un record in BLOCCHI). $SCARICO$ indica quali sono le operazioni di scarico in corso con indicazione dell'utente che fa l'upload ed il download e il blocco in via di trasferimento. L'unità di trasferimento il blocco. $RICHIESTA$ indica le richieste in attesa di scarico, con Time il tempo in cui la richiesta è stata inoltrata.

Esercizio 61 *(punti 8) Si scriva una espressione dell'algebra relazionale che se valutata fornisca lo userid degli utenti che hanno il maggior numero di download in corso.*

Esercizio 62 *(punti 8) Si scriva una vista SQL che per ogni utente fornisca il numero di upload e download in corso, il numero di file in attesa di dowload e il tempo massimo di attesa.*

Esercizio 63 *(punti 10) Quando un utente commuta il suo stato da Off a On i file o i blocchi di file che possiede divengono disponibili. Di conseguenza un trigger cerca di attivare il maggior numero di operazioni di scarico presenti nella tabella CODA usando le disponibilità dell'utente secondo i seguenti criteri: Hanno precedenza le richieste più vecchie; le richieste riguardano file posseduti dall'utente che ha fatto accosso (si osservi che se un utente possiede un intero file mette a disposizione tutti blocchi del file); il numero di operazioni di carico attivate non può superare il MaxUp dell'utente; una richiesta non può essere attivata se l'utente ha già in corso un numero di download pari al numero massimo a lui consentito.*
Attivare una richiesta significa inserire una riga corrispondente nella tabella $SCARICO$ e rimuovere la richiesta dalla tabella CODA. Scrivere il trigger in questione.

Esercizio 64 *(punti 8) Utilizzando SQL dimamico si scriva una funzione il cui scopo è quello di fornire in modo parametrico dei conteggi su file (Idfile). La funzione richiede tre parametri. Il primo è un Idfile, il secondo il nome di una tabella (valori possibili FILE, BLOCCHI, SCARICO, RICHIESTA), il terzo il nome di un attributo della tabella passato nel secondo parametro. La funzione restituisce il valore di una COUNT(DISTINCT) applicata all'attributo passato sul secondo parametro della tabella passata nel primo parametro per l'dentificativo di file passato nel primo parametro. Ad esempio, se vengono passati i tre parametri attuali $Id-50$, $SCARICO$, $USRUP$ la funzione restituisce il valore degli upload distinti (COUNT DISTINCT su UsrUp) sulla tabella $SCARICO$ per lo $IdFile = Id-50$.*

7 Scritto del 28 maggio 2011

Si consideri il seguente schema relazionale che descrive la strutturazione di classi in un class diagram di UML.

$CLASS(\underline{Nome}, Descrizione, DefVal)$
$REFINE(SuperClass, SubClass)$
$ATTRIBUTI(Class, Nome, Tipo)$
$BASICTYPE(Nome, DefVal)$
$OBJ(\underline{Oid}, NomeClass)$
$OBJATT(\underline{Oid, NomeAtt}, ValBasic, ValObj)$.

$CLASS$ descrive le classi: l'attributo DefVal contiene l'oid di una istanza di default per la classe (chiave esterna verso la tabella OBJ). $REFINE$ descrive la relazione di specializzazione delle classi (SubClass è la specializzazione della classe SuperClass); $ATTRIBUTI$ indica quali attributi vengono associati ad una classe. Il tipo di un attributo può essere di tipo basico (contenuto nello schema $BASICTYPE$ dove l'attributo $defVal$ indica il valore di default per il tipo basico) oppure il nome di una classe. OBJ descrive le istanze delle classi (oggetti) con Oid l'identificativo dell'istanza. I valori degli attributi associati agli oggetti sono memorizzati in $OBJATT$: se l'attributo ha tipo basico allora $ValBasic$ è non vuoto (ValObj è invece vuoto) e contiene il valore dell'attributo; se l'attributo ha come tipo una classe allora $ValBasic$ è vuoto e ValObj contiene l'oid di un oggetto.

Esercizio 71 *(7 punti) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce il nome delle classi non specializzate che prevedono solo attributi di tipo BASICTYPE.*

Esercizio 72 *(8 punti) Si scriva una interrogazione SQL che recupera coppie di nomi di classi in modo che per ogni attributo della seconda classe vi sia un attributo della prima classe che abbia lo stesso nome e lo stesso tipo.*

Esercizio 73 *(11 punti) Si scriva un metodo PLSQL che riceve in ingresso il nome di una classe e un nuovo identificativo di oggetto e che crea la istanziazione di un oggetto default per quella classe. L'oggetto creato dovr essere inserito nelle opportune tabelle (OBJ e OBJATT). Per gli attributi di tipo basico si assoceranno agli attributi i valori di default in dicati in BASICTYPE. Per gli attributi di tipo tipo classe si assoceranno agli attributi i valori di default in dicati in CLASS. Si presti attenzione al fatto che se la classe da istanziare risulta essere il raffinamento di un'altra classe anche gli attributi non sovrascritti di tutte le classi antenate (fino alla radice della gerarchia di specializzazione) vanno istanziati.*

Esercizio 74 *(8 punti) Si scriva usando SQL dinamico una procedura PSQL che riceve in ingreasso il nome di una classe e una stringa del formato 'NomeAttributo1@Valoreattributo1@NomeAttributo2@Valoreattributo2@...' (numero arbitrario di coppie attibuto valore attributo). L'effetto della procedura di restituire il numero di oggetti di quella classe per cui per tutte le coppie valga nomeattributo = valoreattributo.*

(si ricorda che in PSQL la funzione SUBSTR(stringa, pos, lungh) restituisce la porzione della sottostringa di stringa che ha inizio in pos ed ha lunghezza lungh e che

INSTR(instringa,stringa,pos) restituisce la prima posizione di instringa in cui compare a partire da pos la prima occorrenza della sottostringa stringa).

8 Scritto del 29 giugno 2011

Si consideri il seguente schema relazionale che descrive un database contenente domande per test a risposta multipla, test e risultati di somministrazione di test.

$QUESITO(\underline{CodQ}, Testo, Tipo, Livello)$
$RISPOSTE(\underline{CodQ, CodR}, Risposta, Ok)$
$TEST(\underline{CodT}, Data, Luogo, PuntE, PuntC, PuntN, Elab)$
$COMPTEST(codT, CodQ)$
$RISULTATO(codT, Utente, CodQ, CodR)$
$VALUTAZIONE(Utente, CodT, Punti)$
$STORICO(Utente, NTest, Punteggio)$

Lo schema $QUESITO$ descrive domande con risposte a scelta multipla. Ogni domanda appartiene ad una tipologia ed ha associato un livello di difficoltà. Le risposte multiple alle domande si trovano in $RISPOSTE$: l'attributo *ok* assume valori $\{0, 1\}$ (0 per la risposta sbagliata e 1 per quella corretta). Ogni domanda ha più risposte e tra queste solo una è corretta. Lo schema $TEST$ descrive i test effettuati (un test è una collezione di domande somministrata a uno o più utenti). PuntE indica i punti per una risposta errata, PuntC i punti per una risposta corretta, PuntN i punti per una risposta non data, Elab un flag che assume valori $\{0, 1\}$ e che indica se i risultati del test sono stati elaborati.

La scelta delle domande presenti in un test è indicata nella tabella $COMPTEST$. Nello schema $RISULTATO$ si ha descrizione della risposta data da un utente per una domanda in un test. Se l'utente non ha fornito risposta ad una domanda presente in un test l'attributo $CodR$ assume valore nullo. La tabella $VALUTAZIONE$ contiene il punteggio conseguito da un utente in un test. La tabella $STORICO$ tiene traccia del numero di test sostenuto da ciascun utente e del punteggio medio conseguito nei test.

Esercizio 81 *(7 punti) Si scriva una interrogazione in algebra relazionale che fornisce coppie codice test - utente relativi a test per i quali l'utente ha fornito tutte le risposte correttamente.*

Esercizio 82 *(7 punti) Si scriva una interrogazione SQL fornisce coppie codice test - utente relativi a test per i quali l'utente ha fornito il maggior numero di risposte corrette (rispetto agli utenti che hanno sostenuto lo stesso test).*

Esercizio 83 *(9 punti) Si scriva un trigger SQL che elabora i risultati del test quando il flag Elab passa da 0 a 1. IL trigger deve calcolare il punteggio conseguito da ciascun utente nel test e memorizzarlo nella tabella $VALUTAZIONE$. La tabella $STORICO$ deve essere aggiornata di conseguenza.*

Esercizio 84 *(9 punti) Si scriva usando SQL dinamico una procedura PSQL che riceve in ingresso il codice di un test e una stringa contenente un numero variabile di codici di domande che debbono comporre il test (i codici di domande sono separati dal carattere @, ad es 'Cod1@Cod2@Cod3...'). La procedura elabora la stringa creando un'unica istruzione INSERT che inserisce le coppie codice test - codice domanda nella tabella COMPTEST e la amnda in esecuzione.*

(si ricorda che in PSQL la funzione SUBSTR(stringa, pos, lungh) restituisce la porzione della sottostringa di stringa che ha inizio in pos ed ha lunghezza lungh e che

INSTR(instringa,stringa,pos) restituisce la prima posizione di instringa in cui compare a partire da pos la prima occorrenza della sottostringa stringa).

9 Scritto del 21 luglio 2011

Si consideri il seguente schema relazionale che descrive un frammento della base di dati per gestire un gioco da schacchiera

$Caselle(\underline{CodC}, X, Y)$
$Inizio(Partita, Pezzo, CodC)$
$Mosse(\underline{Partita, Ordine}, Pezzo, Partenza, Arrivo, Ultima)$
$Stop(Partita, Pezzo, Ordine)$

Caselle descrive le coordinate delle caselle della scacchiera occupabili dalle pedine. *Inizio* tiene traccia della posizione iniziale delle pedine in una partita. *Mosse* memorizza le sequenze (ordinate secondo l'attributo *Ordine*) di mosse in una partita. Per ogni mossa viene indicata la casella di partenza e quella di arrivo. L'attributo ultima (con valori 0,1) indica se la mossa rappresenta l'ultima mossa della partita. $Stop(Partita, Pezzo, Ordine)$ tiene traccia delle pedine eliminate (ordine indica la mossa che ha portao all'eliminazione).

Esercizio 91 *Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce codice di partita e pezzo per pezzi che nella partita hanno fatto solo o mosse orizzontali o mosse verticali.*

Esercizio 92 *Si supponga che oltre agli schemi già elencati vi sia un altro schema riassuntivo per ogni partita con la seguente struttura:*
$Storico(Partita, Pezzo, NMosse, NCaselle)$ *dove* $NMosse$ *indica il numero di mosse del pezzo nella partita e* $NCaselle$ *indica indica il numero di caselle diverse attraversate dal pezzo. Quando viene inserita l'ultima mossa della partita, un trigger aggiorna lo schema Storico con i dati della partita. Scrivere il trigger.*

Esercizio 93 *Si scriva ua procedura PLSQL che prende in ingresso una partita e un numero d'ordine di mossa N e restituisca in uscita la situazione della scacchiera dopo che si sono giocate le prime N mosse della partita. La situazione della scacchiera vine descritta mediante una stringa che contiene una sequenza di terne -X,Y,pezzo- (separate da un carattere separatore) ad indicare quali sono le caselle occupate dai pezzi (le caselle vuote non vengono elencate nella stringa).*

Esercizio 94 *Si scriva una procedura PLSQL che riceve in ingresso il nome per una nuova tabella e una stringa di coppie attributo - tipo (della forma attributo1@tipo1@attributo2@tipo2..... @attributok@tipok). Usando SQL dinamico si scriva un comando (e lo si esegua) per la creazione della tabella che abbia come nome il nome indicato nel primo parametro e come attributi quelli indicati nella stringa passata come secondo parametro. Si ricorda che in Oracle la funzione SUBSTR(stringa, pos, lungh) restituisce la porzione della sottostringa di stringa che ha inizio in pos ed ha lunghezza lungh) e che la funzione INSTR(stringa, search, occ) restituisce la posizione iniziale della occorrenza occ (occ = 1 per la prima occorrenza) della stringa search in stringa se esistente e -1 altrimenti.*

10 Scritto del 3 marzo 2010

Si consideri il seguente schema relazionale che descrive un frammento della base di dati per gestire un sistema di posta elettronica

$User(\underline{email}, Nome, Cognome, Note)$
$Mail(\underline{Idmail}, Timestamp, Stato, Oggetto, Testo, Dimensione)$
$Legame(Idmail, utente, tipo)$
$Folder(\underline{IdFolder}, Utente, Nome, Dimensione, Contenente)$
$Contiene(IdFolder, Idmail)$

User descrive gli utenti noti che possono essere o mittenti o destinatari di una email. *Mail* descrive il messaggio di posta ricevuto: stato indica se il messaggio è stato letto (READ), non è stato letto (UNREAD); dimensione indica la dimensione in byte del messaggio. *Legame* indica gli utenti interessati ad una email; *tipo* indica se l'utente è mittente (FROM), destinatario (TO), per conoscenza (CC); un messaggio ha un solo mittente ma possibilmente più destinatari. *Folder* indica il nome di un folder di un utente dove sono organizzate i messaggi: un forder appartiene ad un utente; fa parte di una organizzazione ad albero (directory/sottodirectory come in un file system); Contenente indica il folder contenete (se il folder è la radice il valore è NULL); Dimensione indica la dimensione cumulata di tutti i messaggi contenuti nel folder e nei suoi sottofolder. *Contiene* associa i messaggi ai folder.

Esercizio 101 *Si scriva una espressione in algebra relazionale che se valutata fornisca coppie di utenti mittente - destinatario tali che l'utente mittente abbia inviato dei messaggi al destinatario ma il destinatario non abbia mail letto nessuno dei messaggi inviati da quel mittente.*

Esercizio 102 *Si scriva una vista che raccolga la seguente informazione aggregata relativa ai messaggi inviati/ricevuti nel solo mese corrente (si usi SYSDATE per la data corrente). Per ogni utente, il numero di messaggi inviati, il numero di messaggi ricevuti, il numero medio di messaggi ricevuti per utente mittente.*

Esercizio 103 *Si scriva un trigger che viene attivato quando viene aggiunto un messaggio ad un folder (tabella Contenente). L'effetto previsto è quello di incrementare la dimensione del folder di una quantità pari alla dimensione del messaggio il folder che lo contiene e la dimensione di tutti i folder (a risalire fino alla radice della gerarchia di contenimento) che lo contengono.*

Esercizio 104 *Si scriva una procedura PLSQL che riceve in ingresso il nome di due tabelle Tab1 e Tab2, il nome di un vincolo Vinc, e una stringa di coppie di attributi di Tab1 e Tab2, rispettivamente, (della forma attr1Tab1@attr1Tab2@attr2Tab1@attr2Tab2..... @attrkTab1@attrkTab2). L'idea è che in Tab2 deva essere inserita un vincolo di chiave esterna per gli attributi (attr1Tab2, ..., attrkTab2) che referenziano la chiave primaria (attr1Tab1, ..., attrkTab1) di Tab1. Usando SQL dinamico si scriva un comando (e lo si esegua) che implementi le richieste di cui sopra. Si ricorda che in Oracle la funzione SUBSTR(stringa, pos, lungh) restituisce la porzione della sottostringa di stringa che ha inizio in pos ed ha lunghezza lungh) e che la funzione INSTR(stringa, search, occ) restituisce la posizione iniziale della occorrenza occ (occ = 1 per la prima occorrenza) della stringa search in stringa se esistente e -1 altrimenti.*

11 Scritto del 24 novembre 2011

Basi di Dati e Sistemi Informativi I, Prove scritte AA 2010/11

Adriano Peron

Facoltà di Scienze M.F.N., Corso di Laurea in Informatica, Dipartimento di Scienze Fisiche,
Università di Napoli 'Federico II', Italy
E-mail: peron@na.infn.it

1 Scritto del 20 dicembre 2010 - Intercorso A

Si consideri il seguente schema relazionale che descrive operazioni di acquisto on-line con carrello (il cliente mette nel carrello gli articoli scelti; alla fine dell'operazione decide se procedere all'ordine indicando che il carrello è completo; quando il carrello è completo il carrello viene trasformato in ordine)
$CLIENTE(\underline{CodCl}, Nome, Cognome, Punti)$
$ORDINI(\underline{CodO}, CodCl, Data)$
$COMPORD(CodO, CodA, Quantita)$
$CARRELLO(CodO, Data, CodCl, Completo)$
$COMPCARRELLO(CodO, CodA, Quantita)$
$ARTICOLO(\underline{CodA}, Costo, Punti, Scorta)$.
$CLIENTE$, fornisce la descrizione del cliente (Punti fornisce i punti cumulati dal cliente per gli acquisti fatti); $ORDINI$ forisce la descrizione di un ordine completato (CodCl indica il codice cliente). $COMPORD$ indica la composizione dell'ordine (Codice dell'ordine, codice dell'articolo e quantità dell'articolo). $CARRELLO$ descrive un ordine non ancora consolidato (Completo è NULL se il carrello non è stato completato e NOT NULL altrimenti). $COMPCARRELLO$ indica la composizione del carrello (Codice dell'ordine, codice dell'articolo e quantità dell'articolo). $ARTICOLO$ descrive gli articoli in vendita (Codice, Costo unitario, Punti guadagnati per ogni singolo acquisto, scorta in magazzino dell'articolo).

Esercizio 11 *Si scriva una espressione in algebra relazionale (senza usare operazioni di conteggio) che, se valutata, fornisce il codice degli utenti che in tutti i loro ordini hanno comperato sempre almeno due articoli.*

Esercizio 12 *Si scriva una interrogazione SQL (preferibilmente senza uso di viste) che, se valutata, fornisce il codice degli utenti che (complessivamente nei vari ordini) hanno comperato tutti gli articoli disponibili.*

Almeno uno dei due seguenti esercizi.

Esercizio 13 *Si fornisca l'implementazione più ragionevole per i seguenti vincoli:*

1. *La scorta di un articolo non deve essere inferiore alla quantità complessiva dell'articolo presente nella composizione dei carrelli (ci possono essere più carrelli);*
2. *Ogni cliente può avere al più un carrello;*
3. *Gli ordini e i carrelli sono associati solo a clienti noti.*

Esercizio 14 *Si supponga che il carrello di codice 'YYY' venga completato. Il completamento del carrello ha il seguente effetto: il carrello e la sua composizione viene trasformato in ordine (conservando il codice) e rimosso dalla tabella dei carrelli; Le scorte di magazzino vengono aggiornate stornando i quantitativi indicati nel carrello. I punti del cliente vengono aggiornati con i punti acquisiti dall'acquisto).*

1. *Si indichi a parole quali operazioni vengo fatte e in che ordine.*
2. *Si scriva in SQL l'operazione per l'aggiornamento della scorta in magazzino.*

2 Scritto del 20 dicembre 2010 - Intercorso B

Si consideri il seguente schema relazionale che descrive operazioni di acquisto on-line con carrello (il cliente mette nel carrello gli articoli scelti; alla fine dell'operazione decide se procedere all'ordine indicando che il carrello è completo; quando il carrello è completo il carrello viene trasformato in ordine)
$CLIENTE(\underline{CodCl}, Nome, Cognome, Punti)$
$ORDINI(\underline{CodO}, CodCl, Data)$
$COMPORD(CodO, CodA, Quantita)$
$CARRELLO(CodO, Data, CodCl, Completo)$
$COMPCARRELLO(CodO, CodA, Quantita)$
$ARTICOLO(\underline{CodA}, Costo, Punti, Scorta)$.
$CLIENTE$, fornisce la descrizione del cliente (Punti fornisce i punti cumulati dal cliente per gli acquisti fatti); $ORDINI$ forisce la descrizione di un ordine completato (CodCl indica il codice cliente). $COMPORD$ indica la composizione dell'ordine (Codice dell'ordine, codice dell'articolo e quantità dell'articolo). $CARRELLO$ descrive un ordine non ancora consolidato (Completo è NULL se il carrello non è stato completato e NOT NULL altrimenti). $COMPCARRELLO$ indica la composizione del carrello (Codice dell'ordine, codice dell'articolo e quantità dell'articolo). $ARTICOLO$ descrive gli articoli in vendita (Codice, Costo unitario, Punti guadagnati per ogni singolo acquisto, scorta in magazzino dell'articolo).

Esercizio 21 *Si scriva una espressione in algebra relazionale (senza usare operazioni di conteggio) che, se valutata, fornisce il codice degli utenti che (complessivamente nei vari ordini) hanno comperato tutti gli articoli disponibili.*

Esercizio 22 *Si scriva una interrogazione SQL (preferibilmente senza uso di viste) che, se valutata, fornisce il codice degli utenti che in tutti i loro ordini hanno comperato sempre almeno due articoli.*

Almeno uno dei due esercizi.

Esercizio 23 *Si fornisca l'implementazione più ragionevole per i seguenti vincoli:*

1. *I punti di un cliente devono esere inferiori o uguali alla somma di tutti i punti cumulati dal cliente negli ordini del 2010;*
2. *Ogni cliente può fare al più un ordine al giorno;*
3. *Le composizioni di ordini e di carrelli sono associati solo ad articoli trattati in magazzino.*

Esercizio 24 *Si supponga che il carrello di codice 'YYY' venga completato. Il completamento del carrello ha il seguente effetto: il carrello e la sua composizione viene trasformato in ordine (conservando il codice) e rimosso dalla tabella dei carrelli; Le scorte di magazzino vengono aggiornate stornando i quantitativi indicati nel carrello. I punti del cliente vengono aggiornati con i punti acquisiti dall'acquisto).*

1. *Si indichi a parole quali operazioni vengo fatte e in che ordine.*
2. *Si scriva in SQL l'operazione per l'aggiornamento del punteggio del cliente.*

3 Scritto del 2 febbraio 2011

Per il compito intero si svolgano i primi quattro esercizi.
Per la seconda prova intercorso si svolgano gli ultimi tre esercizi.

Si consideri il seguente schema relazionale che descrive la strutturazione di un documento nelle sue parti (sezioni, sottosezioni, etc) e l'associazione tra parole chiave e le sezioni dove esse sono presenti.

$SEZIONE(\underline{Documento, CodSezione}, Titolo, Testo)$
$STRUTTURA(\underline{Documento, SezContenente}, SezContenuta, Posizione)$
$PAROLE(Parola)$
$PRESENZE(Parola, Documento, Sezione, Offset)$

$SEZIONE$ descrive le sezioni dei documenti (ogni sezione ha un titolo ed un testo. Una sezione può essere composta da sezioni. Tale composizione è descritta in $STRUTTURA$ dove SezContenente è il codice della sezione composta e SezContenuta è il codice della sottosezione e Posizione è una stringa di caratteri i che rappresenta un intero e che indica che SezContenuta è la i-esima sottosezione di SezContenente.
$PAROLE$ fornisce la lista delle parole interessanti (parole chiave) per il recupero delle sezioni. Se una parola chiave è presente in una sezione, è riportata una riga corrispondente in $PRESENZE$ (Sezione in $PRESENZE$ indica il codice di una sezione e offset indica la posizione della parola nella sezione; si ricorda che in ORACLE la funzione INSTR(stringa, pattern, [inizio, [occorrenza]]) restituisce, se presente, la posizione del pattern nella stringa)

Esercizio 31 *(7 punti) Si scriva una espressione dell'algebra relazionale che se valutata fornisca tutti i documenti in cui sono presenti tutte le parole chiave.*

Esercizio 32 *(7 punti) Si scriva una interrogazione SQL che che per ogni documento fornisca tutte le coppie di parole chiavi adiacenti nel documento (si trovano nella stessa sezione e non sono separate da altra parola chiave).*

Esercizio 33 *(7 punti) Si scriva una funzione PLSQL che riceve in ingresso il codice di due documenti e che restituisce un intero che rappresenta il numero di parole che i due documenti hanno in comune.*
Si utilizzi la funzione definita per scrivere una vista SIMILITUDINE(Doc1,Doc2,NumParoleComuni) che per ogni coppia di documenti Doc1 e Doc2 riporta il numero di parole in comune.

Esercizio 34 *(6 punti un trigger, 9 due trigger) Si scriva almeno uno dei due trigger seguenti. Si scriva un trigger che quando viene inserita una nuova parola chiave nella tabella $PAROLA$ provveda ad aggiornare la tabella $PRESENZE$ con la indicazione dei documenti, delle sezioni che contengono la parola (e i relativi offset). Si scriva inoltre un trigger che quando viene inserita una nuova sezione provveda ad aggiornare la tabella $PRESENZE$ relativamente a tutte le parole chiave presenti nella nuova sezione.*

Esercizio 35 *(Esercizio facoltativo, 10 punti) Si scriva una procedura PLSQL che riceve in ingresso una stringa di parole chiave del formato @parola1@parola2@... @parolak@. Usando SQL dinamico si scriva una interrogazione che trova i documenti che contengono tutte le parole indicate nella stringa (si ricorda che in SQL la funzione SUBSTR(stringa, pos, lungh) restituisce la porzione della sottostringa di stringa che*

ha inizio in pos ed ha lunghezza lungh). Il risultato della interrogazione deve essere restituito in una stringa.

4 Scritto del 2 febbraio 2011 - bis

Per il compito intero si svolgano i primi quattro esercizi.
Per la seconda prova intercorso si svolgano gli ultimi tre esercizi.

Si consideri il seguente schema relazionale che descrive la strutturazione di un documento nelle sue parti (sezioni, sottosezioni, etc) e l'associazione tra parole chiave e le sezioni dove esse sono presenti.

$SEZIONE(\underline{Documento, CodSezione}, Titolo, Testo)$
$STRUTTURA(\underline{Documento, SezContenente}, SezContenuta, Posizione)$
$PAROLE(Parola)$
$PRESENZE(Parola, Documento, Sezione, Offset)$

$SEZIONE$ descrive le sezioni dei documenti (ogni sezione ha un titolo ed un testo. Una sezione può essere composta da sezioni. Tale composizione è descritta in $STRUTTURA$ dove SezContenente è il codice della sezione composta e SezContenuta è il codice della sottosezione e Posizione è una stringa di caratteri i che rappresenta un intero e che indica che SezContenuta è la i-esima sottosezione di SezContenente.
$PAROLE$ fornisce la lista delle parole interessanti (parole chiave) per il recupero delle sezioni. Se una parola chiave è presente in una sezione, è riportata una riga corrispondente in $PRESENZE$ (Sezione in $PRESENZE$ indica il codice di una sezione e offset indica la posizione della parola nella sezione; si ricorda che in ORACLE la funzione INSTR(stringa, pattern, [inizio, [occorrenza]]) restituisce, se presente, la posizione del pattern nella stringa)

Esercizio 41 *(10 punti) Si scriva una funzione PLSQL che riceve in ingresso il codice di due documenti e che restituisce un intero che rappresenta il numero di parole che i due documenti hanno in comune.*
Si utilizzi la funzione definita per scrivere una vista SIMILITUDINE(Doc1,Doc2,NumParoleComuni) che per ogni coppia di documenti Doc1 e Doc2 riporta il numero di parole in comune.

Esercizio 42 *(10 punti un trigger, 18 due trigger) Si scriva almeno uno dei due trigger seguenti. Si scriva un trigger che quando viene inserita una nuova parola chiave nella tabella $PAROLA$ provveda ad aggiornare la tabella $PRESENZE$ con la indicazione dei documenti, delle sezioni che contengono la parola (e i relativi offset). Si scriva inoltre un trigger che quando viene inserita una nuova sezione provveda ad aggiornare la tabella $PRESENZE$ relativamente a tutte le parole chiave presenti nella nuova sezione.*

Esercizio 43 *(Esercizio facoltativo, 15 punti) Si scriva una procedura PLSQL che riceve in ingresso una stringa di parole chiave del formato @parola1@parola2@... @parolak@. Usando SQL dinamico si scriva una interrogazione che trova i documenti che contengono tutte le parole indicate nella stringa (si ricorda che in SQL la funzione SUBSTR(stringa, pos, lungh) restituisce la porzione della sottostringa di stringa che ha inizio in pos ed ha lunghezza lungh). Il risultato della interrogazione deve essere restituito in una stringa.*

5 Scritto del 2 marzo 2011

Si consideri il seguente schema relazionale che descrive una versione elementare dei dati per un social network
$Utenti(\underline{id}, nome, cognome, mail, passwd)$
$Amici(Utente1, Utente2, Data)$
$Interessi(Utente, Categoria, Valore)$
$Richiesta(Richiedente, Invitato, Tipo, Data, Stato)$
$Post(\underline{codice}, Utente, testo, data)$
$Commento(codicePost, Utente, testo, data)$.

Amici descrive la relazione simmetrica di amicizia tra utenti e la data in cui è stata stabilita; *Interessi* descrive gli interessi di un utente mediante la categoria di interesse (ad es. Sport) e il tipo di interesse (ad es. valore tennis). *Richiesta* descrive le richieste inoltrate dall'utente Richiedente all'utente Invitato, il tipo richiesta (ad. es Amicizia), lo stato della richiesta (attesa,accettata); un elemento di *Post* indica un elemento testuale pubblicato da un utente; un elemento di *Commento* è la risposta di un utente al post individuato da codicePost.

Esercizio 51 *(8 punti) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce il nome degli utenti che hanno scritto post che hanno ricevuto sempre almeno 5 commenti.*

Esercizio 52 *(8 punti) Si implementino in modo opportuno i seguenti vincoli:*

1. *quando una richiesta di amicizia viene accettata la relazione Amici viene aggiornata stabilendo in modo simmetrico la relazione di amicizia.*
2. *Un utente non può commentare più di una volta lo stesso post;*
3. *Lo stato di una richiesta può avere solo o valore Attesa o valore Accettato o valore Rifiutato;*
4. *Non ci può essere una relazione di amicizia tra due utenti senza una richiesta di amicizia accettata.*

Esercizio 53 *(10 punti) Si scriva una funzione che prende in ingresso il codice di un utente ed un numero intero k. La funzione deve recuperare le catene di amici fino al grado k (di grado 1 sono gli amici ufficiali, di grado 2 gli amici degli amici ect.). Ci si avvalga di una tabella temporanea TMP che si suppone già creata. Una volta recuperati gli amici la funzione deve restituirli in una stringa in ordine crescente di grado di amicizia e, per gli amici dello stesso grado, in ordine lessicografico per cognome e nome.*

Esercizio 54 *(10 punti) Usando sql dinamico si scriva una funzione che riceve in ingresso una stringa di lunghezza variabile di caratteri del tipo ?categoria1?valore1?categoria2?valore2?.... e che restituisca in una stringa gli identificativi degli utenti separati da virgole che hanno interessi che soddisfano almeno una delle coppie ?categoria?valore? (si usino le funzioni INSTR e SUBSTR per la scansione della lista di ingresso).*

6 Scritto del 25 marzo 2011

Si consideri il seguente schema relazionale che descrive un database per la gestione di condivisione di file in uno schema peer to peer.
$UTENTE(\underline{UsrId}, MaxUp, MaxDown, Stato)$
$FILE(IdFile, IdUtente)$
$BLOCCHI(IdFile, IdUtente, IdBlocco)$
$SCARICO(IdFile, UsrUp, UsrDown, IdBlocco)$
$RICHIESTA(IdFile, UsrDown, Time, IdBlocco)$

$UTENTE$ formisce il numero massimo di operazioni di Upload e Download simultaneamente possibili per un utente e il suo stato che può assumere lo stato On o Off. $FILE$ indica i file completi di cui gli utenti sono in possesso. Un utente può essere in possesso di un file completo (in questo caso si ha un record in FILE) oppure può essere in possesso solo di alcuni blocchi del file e non di tutti (in questo caso per ogni blocco posseduto si ha un record in BLOCCHI). $SCARICO$ indica quali sono le operazioni di scarico in corso con indicazione dell'utente che fa l'upload ed il download e il blocco in via di trasferimento. L'unità di trasferimento il blocco. $RICHIESTA$ indica le richieste in attesa di scarico, con Time il tempo in cui la richiesta è stata inoltrata.

Esercizio 61 *(punti 8) Si scriva una espressione dell'algebra relazionale che se valutata fornisca lo userid degli utenti che hanno il maggior numero di download in corso.*

Esercizio 62 *(punti 8) Si scriva una vista SQL che per ogni utente fornisca il numero di upload e download in corso, il numero di file in attesa di dowload e il tempo massimo di attesa.*

Esercizio 63 *(punti 10) Quando un utente commuta il suo stato da Off a On i file o i blocchi di file che possiede divengono disponibili. Di conseguenza un trigger cerca di attivare il maggior numero di operazioni di scarico presenti nella tabella CODA usando le disponibilità dell'utente secondo i seguenti criteri: Hanno precedenza le richieste più vecchie; le richieste riguardano file posseduti dall'utente che ha fatto accosso (si osservi che se un utente possiede un intero file mette a disposizione tutti blocchi del file); il numero di operazioni di carico attivate non può superare il MaxUp dell'utente; una richiesta non può essere attivata se l'utente ha già in corso un numero di download pari al numero massimo a lui consentito.*
Attivare una richiesta significa inserire una riga corrispondente nella tabella $SCARICO$ e rimuovere la richiesta dalla tabella CODA. Scrivere il trigger in questione.

Esercizio 64 *(punti 8) Utilizzando SQL dimamico si scriva una funzione il cui scopo è quello di fornire in modo parametrico dei conteggi su file (Idfile). La funzione richiede tre parametri. Il primo è un Idfile, il secondo il nome di una tabella (valori possibili FILE, BLOCCHI, SCARICO, RICHIESTA), il terzo il nome di un attributo della tabella passato nel secondo parametro. La funzione restituisce il valore di una COUNT(DISTINCT) applicata all'attributo passato sul secondo parametro della tabella passata nel primo parametro per l'dentificativo di file passato nel primo parametro. Ad esempio, se vengono passati i tre parametri attuali $Id-50$, $SCARICO$, $USRUP$ la funzione restituisce il valore degli upload distinti (COUNT DISTINCT su UsrUp) sulla tabella $SCARICO$ per lo $IdFile = Id-50$.*

7 Scritto del 28 maggio 2011

Si consideri il seguente schema relazionale che descrive la strutturazione di classi in un class diagram di UML.

$CLASS(\underline{Nome}, Descrizione, DefVal)$
$REFINE(SuperClass, SubClass)$
$ATTRIBUTI(Class, Nome, Tipo)$
$BASICTYPE(Nome, DefVal)$
$OBJ(\underline{Oid}, NomeClass)$
$OBJATT(\underline{Oid, NomeAtt}, ValBasic, ValObj)$.

$CLASS$ descrive le classi: l'attributo DefVal contiene l'oid di una istanza di default per la classe (chiave esterna verso la tabella OBJ). $REFINE$ descrive la relazione di specializzazione delle classi (SubClass è la specializzazione della classe SuperClass); $ATTRIBUTI$ indica quali attributi vengono associati ad una classe. Il tipo di un attributo può essere di tipo basico (contenuto nello schema $BASICTYPE$ dove l'attributo $defVal$ indica il valore di default per il tipo basico) oppure il nome di una classe. OBJ descrive le istanze delle classi (oggetti) con Oid l'identificativo dell'istanza. I valori degli attributi associati agli oggetti sono memorizzati in $OBJATT$: se l'attributo ha tipo basico allora $ValBasic$ è non vuoto (ValObj è invece vuoto) e contiene il valore dell'attributo; se l'attributo ha come tipo una classe allora $ValBasic$ è vuoto e ValObj contiene l'oid di un oggetto.

Esercizio 71 *(7 punti) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce il nome delle classi non specializzate che prevedono solo attributi di tipo BASICTYPE.*

Esercizio 72 *(8 punti) Si scriva una interrogazione SQL che recupera coppie di nomi di classi in modo che per ogni attributo della seconda classe vi sia un attributo della prima classe che abbia lo stesso nome e lo stesso tipo.*

Esercizio 73 *(11 punti) Si scriva un metodo PLSQL che riceve in ingresso il nome di una classe e un nuovo identificativo di oggetto e che crea la istanziazione di un oggetto default per quella classe. L'oggetto creato dovr essere inserito nelle opportune tabelle (OBJ e OBJATT). Per gli attributi di tipo basico si assoceranno agli attributi i valori di default in dicati in BASICTYPE. Per gli attributi di tipo tipo classe si assoceranno agli attributi i valori di default in dicati in CLASS. Si presti attenzione al fatto che se la classe da istanziare risulta essere il raffinamento di un'altra classe anche gli attributi non sovrascritti di tutte le classi antenate (fino alla radice della gerarchia di specializzazione) vanno istanziati.*

Esercizio 74 *(8 punti) Si scriva usando SQL dinamico una procedura PSQL che riceve in ingreasso il nome di una classe e una stringa del formato 'NomeAttributo1@Valoreattributo1@NomeAttributo2@Valoreattributo2@...' (numero arbitrario di coppie attibuto valore attributo). L'effetto della procedura di restituire il numero di oggetti di quella classe per cui per tutte le coppie valga nomeattributo = valoreattributo.*

(si ricorda che in PSQL la funzione SUBSTR(stringa, pos, lungh) restituisce la porzione della sottostringa di stringa che ha inizio in pos ed ha lunghezza lungh e che

INSTR(instringa,stringa,pos) restituisce la prima posizione di instringa in cui compare a partire da pos la prima occorrenza della sottostringa stringa).

8 Scritto del 29 giugno 2011

Si consideri il seguente schema relazionale che descrive un database contenente domande per test a risposta multipla, test e risultati di somministrazione di test.

$QUESITO(\underline{CodQ}, Testo, Tipo, Livello)$
$RISPOSTE(\underline{CodQ, CodR}, Risposta, Ok)$
$TEST(\underline{CodT}, Data, Luogo, PuntE, PuntC, PuntN, Elab)$
$COMPTEST(codT, CodQ)$
$RISULTATO(codT, Utente, CodQ, CodR)$
$VALUTAZIONE(Utente, CodT, Punti)$
$STORICO(Utente, NTest, Punteggio)$

Lo schema $QUESITO$ descrive domande con risposte a scelta multipla. Ogni domanda appartiene ad una tipologia ed ha associato un livello di difficoltà. Le risposte multiple alle domande si trovano in $RISPOSTE$: l'attributo *ok* assume valori $\{0, 1\}$ (0 per la risposta sbagliata e 1 per quella corretta). Ogni domanda ha più risposte e tra queste solo una è corretta. Lo schema $TEST$ descrive i test effettuati (un test è una collezione di domande somministrata a uno o più utenti). PuntE indica i punti per una risposta errata, PuntC i punti per una risposta corretta, PuntN i punti per una risposta non data, Elab un flag che assume valori $\{0, 1\}$ e che indica se i risultati del test sono stati elaborati.

La scelta delle domande presenti in un test è indicata nella tabella $COMPTEST$. Nello schema $RISULTATO$ si ha descrizione della risposta data da un utente per una domanda in un test. Se l'utente non ha fornito risposta ad una domanda presente in un test l'attributo $CodR$ assume valore nullo. La tabella $VALUTAZIONE$ contiene il punteggio conseguito da un utente in un test. La tabella $STORICO$ tiene traccia del numero di test sostenuto da ciascun utente e del punteggio medio conseguito nei test.

Esercizio 81 *(7 punti) Si scriva una interrogazione in algebra relazionale che fornisce coppie codice test - utente relativi a test per i quali l'utente ha fornito tutte le risposte correttamente.*

Esercizio 82 *(7 punti) Si scriva una interrogazione SQL fornisce coppie codice test - utente relativi a test per i quali l'utente ha fornito il maggior numero di risposte corrette (rispetto agli utenti che hanno sostenuto lo stesso test).*

Esercizio 83 *(9 punti) Si scriva un trigger SQL che elabora i risultati del test quando il flag Elab passa da 0 a 1. IL trigger deve calcolare il punteggio conseguito da ciascun utente nel test e memorizzarlo nella tabella $VALUTAZIONE$. La tabella $STORICO$ deve essere aggiornata di conseguenza.*

Esercizio 84 *(9 punti) Si scriva usando SQL dinamico una procedura PSQL che riceve in ingresso il codice di un test e una stringa contenente un numero variabile di codici di domande che debbono comporre il test (i codici di domande sono separati dal carattere @, ad es 'Cod1@Cod2@Cod3...'). La procedura elabora la stringa creando un'unica istruzione INSERT che inserisce le coppie codice test - codice domanda nella tabella COMPTEST e la amnda in esecuzione.*

(si ricorda che in PSQL la funzione SUBSTR(stringa, pos, lungh) restituisce la porzione della sottostringa di stringa che ha inizio in pos ed ha lunghezza lungh e che

INSTR(instringa,stringa,pos) restituisce la prima posizione di instringa in cui compare a partire da pos la prima occorrenza della sottostringa stringa).

9 Scritto del 21 luglio 2011

Si consideri il seguente schema relazionale che descrive un frammento della base di dati per gestire un gioco da schacchiera

$Caselle(\underline{CodC}, X, Y)$
$Inizio(Partita, Pezzo, CodC)$
$Mosse(\underline{Partita, Ordine}, Pezzo, Partenza, Arrivo, Ultima)$
$Stop(Partita, Pezzo, Ordine)$

Caselle descrive le coordinate delle caselle della scacchiera occupabili dalle pedine. *Inizio* tiene traccia della posizione iniziale delle pedine in una partita. *Mosse* memorizza le sequenze (ordinate secondo l'attributo *Ordine*) di mosse in una partita. Per ogni mossa viene indicata la casella di partenza e quella di arrivo. L'attributo ultima (con valori 0,1) indica se la mossa rappresenta l'ultima mossa della partita. $Stop(Partita, Pezzo, Ordine)$ tiene traccia delle pedine eliminate (ordine indica la mossa che ha portao all'eliminazione).

Esercizio 91 *Si scriva una interrogazione in algebra relazionale che, se valutata, fornisce codice di partita e pezzo per pezzi che nella partita hanno fatto solo o mosse orizzontali o mosse verticali.*

Esercizio 92 *Si supponga che oltre agli schemi già elencati vi sia un altro schema riassuntivo per ogni partita con la seguente struttura:*
$Storico(Partita, Pezzo, NMosse, NCaselle)$ *dove* $NMosse$ *indica il numero di mosse del pezzo nella partita e* $NCaselle$ *indica indica il numero di caselle diverse attraversate dal pezzo. Quando viene inserita l'ultima mossa della partita, un trigger aggiorna lo schema Storico con i dati della partita. Scrivere il trigger.*

Esercizio 93 *Si scriva ua procedura PLSQL che prende in ingresso una partita e un numero d'ordine di mossa N e restituisca in uscita la situazione della scacchiera dopo che si sono giocate le prime N mosse della partita. La situazione della scacchiera vine descritta mediante una stringa che contiene una sequenza di terne -X,Y,pezzo- (separate da un carattere separatore) ad indicare quali sono le caselle occupate dai pezzi (le caselle vuote non vengono elencate nella stringa).*

Esercizio 94 *Si scriva una procedura PLSQL che riceve in ingresso il nome per una nuova tabella e una stringa di coppie attributo - tipo (della forma attributo1@tipo1@attributo2@tipo2..... @attributok@tipok). Usando SQL dinamico si scriva un comando (e lo si esegua) per la creazione della tabella che abbia come nome il nome indicato nel primo parametro e come attributi quelli indicati nella stringa passata come secondo parametro. Si ricorda che in Oracle la funzione SUBSTR(stringa, pos, lungh) restituisce la porzione della sottostringa di stringa che ha inizio in pos ed ha lunghezza lungh) e che la funzione INSTR(stringa, search, occ) restituisce la posizione iniziale della occorrenza occ (occ = 1 per la prima occorrenza) della stringa search in stringa se esistente e -1 altrimenti.*

10 Scritto del 3 marzo 2010

Si consideri il seguente schema relazionale che descrive un frammento della base di dati per gestire un sistema di posta elettronica

$User(\underline{email}, Nome, Cognome, Note)$
$Mail(\underline{Idmail}, Timestamp, Stato, Oggetto, Testo, Dimensione)$
$Legame(Idmail, utente, tipo)$
$Folder(\underline{IdFolder}, Utente, Nome, Dimensione, Contenente)$
$Contiene(IdFolder, Idmail)$

User descrive gli utenti noti che possono essere o mittenti o destinatari di una email. *Mail* descrive il messaggio di posta ricevuto: stato indica se il messaggio è stato letto (READ), non è stato letto (UNREAD); dimensione indica la dimensione in byte del messaggio. *Legame* indica gli utenti interessati ad una email; *tipo* indica se l'utente è mittente (FROM), destinatario (TO), per conoscenza (CC); un messaggio ha un solo mittente ma possibilmente più destinatari. *Folder* indica il nome di un folder di un utente dove sono organizzate i messaggi: un forder appartiene ad un utente; fa parte di una organizzazione ad albero (directory/sottodirectory come in un file system); Contenente indica il folder contenete (se il folder è la radice il valore è NULL); Dimensione indica la dimensione cumulata di tutti i messaggi contenuti nel folder e nei suoi sottofolder. *Contiene* associa i messaggi ai folder.

Esercizio 101 *Si scriva una espressione in algebra relazionale che se valutata fornisca coppie di utenti mittente - destinatario tali che l'utente mittente abbia inviato dei messaggi al destinatario ma il destinatario non abbia mail letto nessuno dei messaggi inviati da quel mittente.*

Esercizio 102 *Si scriva una vista che raccolga la seguente informazione aggregata relativa ai messaggi inviati/ricevuti nel solo mese corrente (si usi SYSDATE per la data corrente). Per ogni utente, il numero di messaggi inviati, il numero di messaggi ricevuti, il numero medio di messaggi ricevuti per utente mittente.*

Esercizio 103 *Si scriva un trigger che viene attivato quando viene aggiunto un messaggio ad un folder (tabella Contenente). L'effetto previsto è quello di incrementare la dimensione del folder di una quantità pari alla dimensione del messaggio il folder che lo contiene e la dimensione di tutti i folder (a risalire fino alla radice della gerarchia di contenimento) che lo contengono.*

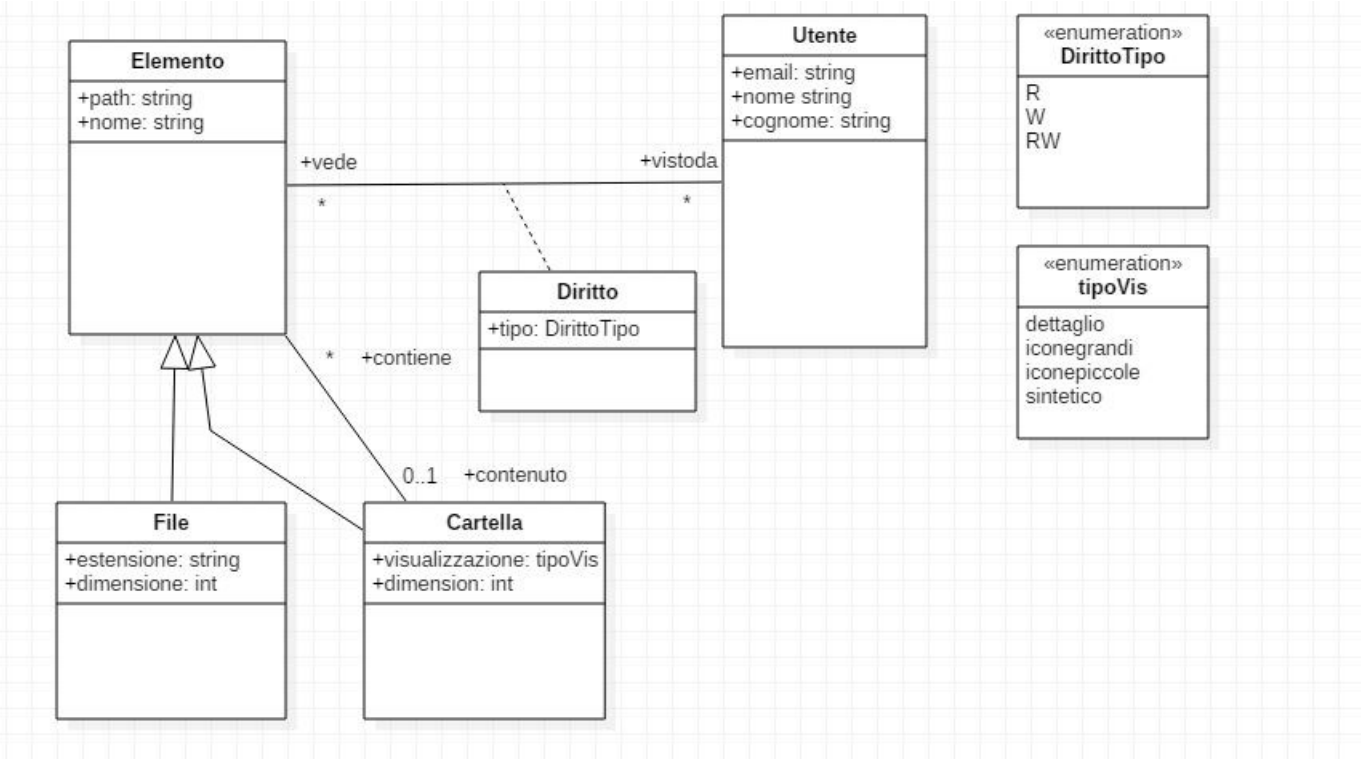
Esercizio 104 *Si scriva una procedura PLSQL che riceve in ingresso il nome di due tabelle Tab1 e Tab2, il nome di un vincolo Vinc, e una stringa di coppie di attributi di Tab1 e Tab2, rispettivamente, (della forma attr1Tab1@attr1Tab2@attr2Tab1@attr2Tab2..... @attrkTab1@attrkTab2). L'idea è che in Tab2 deva essere inserita un vincolo di chiave esterna per gli attributi (attr1Tab2, ..., attrkTab2) che referenziano la chiave primaria (attr1Tab1, ..., attrkTab1) di Tab1. Usando SQL dinamico si scriva un comando (e lo si esegua) che implementi le richieste di cui sopra. Si ricorda che in Oracle la funzione SUBSTR(stringa, pos, lungh) restituisce la porzione della sottostringa di stringa che ha inizio in pos ed ha lunghezza lungh) e che la funzione INSTR(stringa, search, occ) restituisce la posizione iniziale della occorrenza occ (occ = 1 per la prima occorrenza) della stringa search in stringa se esistente e -1 altrimenti.*

11 Scritto del 24 novembre 2011

Basi di dati e sistemi informativi I – 12 novembre 2015

Esercizio 1 Si consideri il class diagram riportato in figura che descrive la strutturazione ad albero di un file system usuale: un elemento può essere o un file o una cartella; una cartella contiene un insieme di file e cartelle.
Gli utenti hanno diritti di accesso esplicitati dalla associazione agli elementi del file system in lettura, scrittura, lettura e scrittura.

1. Si fornisca una ristrutturazione del class diagram per renderlo adeguato ad una traduzione nel modello dei dati relazionale.
2. Si forniscano gli schemi relazionali per il class diagram ristrutturato
3. Si definisca usando SQL la tabella per il folder insieme ai vincoli in essa definibili.



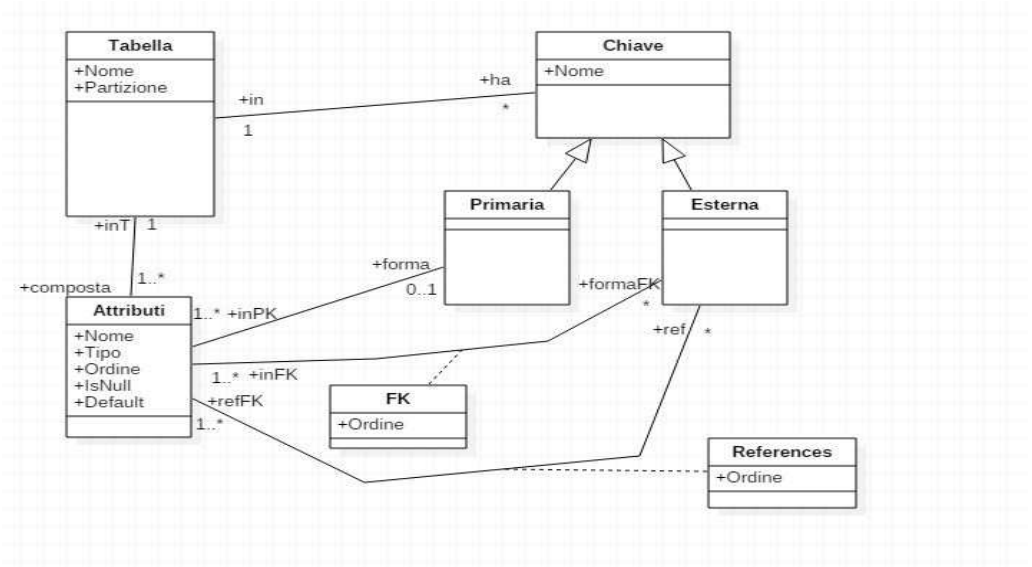
Esercizio N. 2 Si progetti un database per dei cruciverba. Il database dovrà contenere sia la griglia (dimensioni e collocazione delle caselle nere) sia la definizione delle domande verticali ed orizzontali insieme alle loro risposte corrette.
Il DB permette inoltre di memorizzare utenti e le soluzioni da essi forniti ai cruciverba.

Esercizio N. 3 Algebra relazionale

Basi di dati e sistemi informativi I – 23 novembre 2015
Prova A

Esercizio 1 (Punti 12) Si consideri il class diagram riportato in figura che descrive la strutturazione di un insieme di tabelle relazionali formate da attributi e vincoli di chiave primaria ed esterna (l'associazione con classe FK descrive l'insieme di attributi che formano al chiave e l'associazione con classe References descrive l'insieme di attributi riferiti dalla chiave esterna).

1. Si fornisca una ristrutturazione del class diagram per renderlo adeguato ad una traduzione nel modello dei dati relazionale.
2. Si forniscano gli schemi relazionali per il class diagram ristrutturato
3. Si definisca usando SQL le tabelle per attributi e chiave.
4. Nella definizione della tabella per Attributi si scrivano vincoli che assicurino che il valore IsNull possa avere solo valori S o N; che in una stessa tabella non vi possano essere due attributi con lo stesso nome; che due attributi nella stessa tabella non possano avere lo stesso numero d'ordine.



Esercizio N. 2 (Punti 10) Si consideri il seguente schema relazionale che descrive un frammento della base di dati per gestire i prestiti in una biblioteca. Di un libro (descrizione del libro) possono esserci più copie fisiche (esemplari). L'attributo booleano Prestito indica se l'esemplare può essere prestato. L'attributo booleano Consultazione indica se l'esemplare può essere consultato in loco. Il prestito riguarda le copie fisiche, la prenotazione riguarda il libro: quando un esemplare del libro è disponibile si può effettuare il prestito. Ogni utente ha un profilo che regola la durata dei prestiti e il massimo numero di esemplari che può avere in prestito. Un prestito ha una data di effettuazione, una data di scadenza e una data di restituzione (che può essere NULL se il libro non è stato ancora restituito), una data di sollecito (che può essere NULL se il prestito non è ancora scaduto.)

LIBRO(ISBN,Titolo,Editore,Anno)
ESEMPLARE(ISBN,CodiceBarre,Collocazione,Prestito,Consultazione)
UTENTE(CF,CodProfilo,Nome,Cognome,DataN)
PROFILO(CodProfilo,MaxDurata,MaxPrestito)
PRESTITI(CodPrestito,CodiceBarre,Utente,Data,Scadenza,Restituzione,Sollecito)
PRENOTAZIONE(CodPrenotazione,ISBN,Utente,Data)

Si scriva una interrogazione in SQL che restituisca CF, nome e cognome di utenti che in ogni anno hanno preso in prestito lo stesso numero di libri (si usi la funzione YEAR su una data per estrarre l'anno della data) .

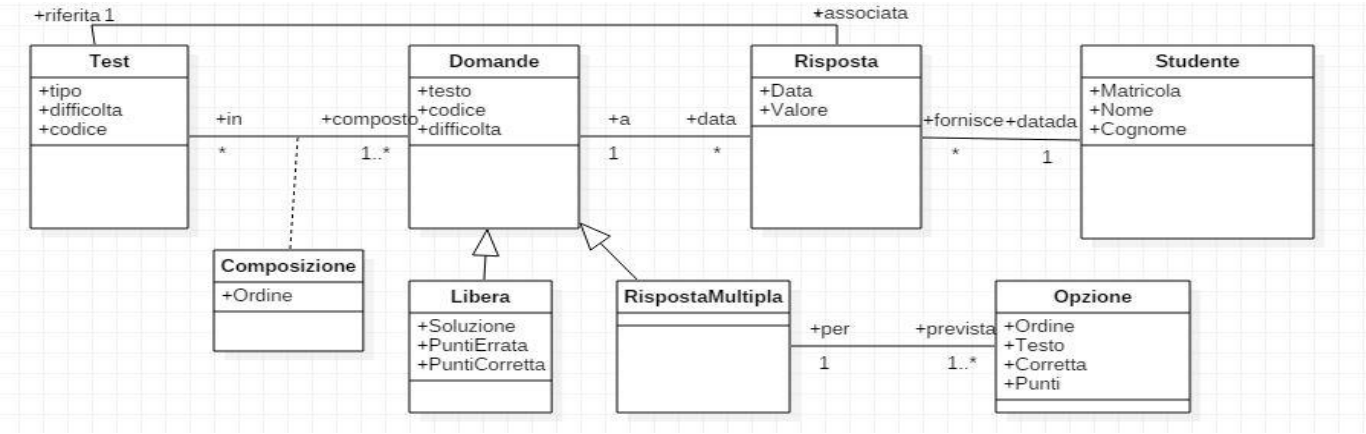
Esercizio N. 3 (Punti 10) Si fornisca un Class Diagram **di progettazione** per il problema descritto nell'esercizion N.2. Il Class diagram deve estendere i dati già modellati nello scema logico fornito permettendo di

- descrivere gli autori dei libri
- documentare i solleciti inviati per prestiti scaduti immaginando che per un prestito scaduto ci possano essere più solleciti consecutivi.
- Inserire informazioni relative a danni che il libro ha subito durante i prestiti, furti o smarrimenti.

Basi di dati e sistemi informativi I – 23 novembre 2015
Prova B

Esercizio 1 (Punti 12) Si consideri il class diagram riportato in figura che descrive la gestione di test per l'apprendimento. I test possono contenere domande a risposta multipla e domande a risposta libera. Per le domande a risposta multipla sono presenti le diverse opzioni e l'indicazione di quale sia corretta. Oltre al test vengono memorizzate le risposte che gli studenti danno alle domande.

1. Si fornisca una ristrutturazione del class diagram per renderlo adeguato ad una traduzione nel modello dei dati relazionale.
2. Si forniscano gli schemi relazionali per il class diagram ristrutturato
3. Si definisca usando SQL le tabelle per le domande e le opzioni di risposta.
4. Nella definizione della tabella per le domande si scrivano vincoli che assicurino che il valore della difficoltà della domanda sia compreso 1 e 7; che due opzioni per la stessa risposta non possano avere lo stesso numero d'ordine.



Esercizio N. 2 (Punti 10) Si considerino gli schemi relazionali di seguito riportati che descrivon un frammento del database per lo stradario e l'anagrafe di una città. Lo spazio urbano è suddiviso in quartieri. I quartieri sono ripartiti in isolati in isolati. Un isolato è una area delimitate da segmenti di vie. Una via ha più segmenti e può delimitare più isolati. Ogni segmento tiene traccia della parià dei numeri civici (numeri pari o numeri dispari) e del numero civico massimo e minimo presente nel segmento.

QUARTIERE(<u>CodQ</u>,Nome,Descrizione)
ISOLATO(<u>CodI</u>, CodQ, Nome)
VIA(<u>CodV</u>,Nome)
SEGMENTO(<u>CodS</u>, CodV, CodI, Pari, NMin,NMax)
RESIDENTI(<u>CF</u>, Nome, Cognome, CodV, Numero)

Segmento contiene la chiave esterna alla Via e all Isolato. L' attributo Pari è un valore booleano che indica se nel segmento di via considerato vi siano numeri civici Pari o Dispari. Una persona risiede in un segmento di via se il numero civico di residenza è compreso tra il valore minimo e massimo del segmento.
Si scriva una interrogazione di algebra relazionale che restituisca gli isolati in cui OGNI segmento dell'isolato ha lo stesso numero di residenti.

Esercizio N. 3 (Punti 10) Si fornisca un Class Diagram **DI PROGGETTAZIONE** per il problema descritto nell'esercizion N.2. Il Class diagram deve estendere i dati già modellati nello scema logico fornito permettendo di

- tenere traccia del fatto che una persona può cambiare residenza e che dunque la residenza ha un inizio, una fine e una persona può avere nel tempo più residenze;
- Tenere traccia dei nuclei familiari (insiemi di persone che vivono insieme nella stessa unità abitativa associata ad un numero civico);
- tener traccia dei legami marito – moglie , e genitore figlio dei residenti.