DATABASE

Domanda 1)
Con riferimento alle basi di dati distribuite, illustrare il funzionamento del protocollo *two-phase commit*.
Illustrare dapprima il funzionamento in *assenza di guasti* e poi descrivere cosa succede in caso di *guasti* e le *azioni di ripristino* necessarie.

TWO PHASE commit "protocollo di aggiornamento a due fasi"
algoritmo distribuito che comprende tutti i nodi in un sistema distribuito per **convalidare una transazione.**
USO: aggiornamenti dei dati necessitano di avvenire **simultaneamente in DB multipli** all'interno di un sistema distribuito.
Garantisce **INTEGRITA'**, ACCURATEZZA dati attraverso bloccaggi sincronizzati "LOCK". "utilizzata carte di credito, aerei…"

RISPOSTA:
TM Gestore delle transazioni (coordinatore)
RM Gestori di risorse (server , gruppi)

- TM
  - scrive il record `prepare` nel suo log e manda un messaggio `prepare` a tutti gli RM; imposta timeout
- ogni RM
  - se in stato affidabile: scrive nel log il record `ready` e trasmette al TM il messaggio `ready`, che indica la scelta di partecipare al protocollo
  - se in stato non affidabile: manda un messaggio `non-ready` e termina la propria partecipazione al protocollo
- TM
  - raccoglie i messaggi di risposta e scrive log
    - `global commit` se tutti gli RM hanno risposto `ready`
    - `global abort` se almeno un RM ha risposto `non-ready` o timeout scattato e non tutti i messaggi ricevuti

- TM
  - trasmette la sua decisione globale (`commit` o `abort`) a tutti gli RM; imposta timeout
- ogni RM in stato `ready`
  - scrive nel log il record relativo alla decisione globale e manda un acknowledgement (`ack`) al TM
  - esegue in loco decisione globale
- TM
  - raccoglie tutti gli `ack` dagli RM coinvolti nella seconda fase
  - se timeout scade, stabilisce un altro timeout e ripete messaggio a tutti gli RM dai quali non ha ricevuto ack
  - quando tutti gli ack sono arrivati, scrive il record `complete` nel suo log

PROTOCOLLI DI RIPRISTINO:

Caduta di un RM

Ripresa a caldo, dipende da ultimo record di log
- `abort` o azione: undo della transazione
- `commit`: redo della transazione
- `ready`: guasto successo durante il two-phase commit; bisogna chiedere al TM

Durante la ripresa a caldo:
- gli identificatori delle transazioni in dubbio sono raccolte in un insieme READY
- per ognuna di queste la decisione finale deve essere richiesta al TM

Caduta del TM

Ripresa a caldo, dipende da ultimo record di log
- `prepare`: alcuni RM potrebbero essere in uno stato di blocco; due opzioni
  - scrivere un `global abort` nel log, e eseguire la seconda fase del protocollo
  - ripetere la prima fase
- `global commit/abort`: alcuni RM potrebbero essere stati correttamente informati mentre altri potrebbero essere ancora bloccati
  - TM deve ripetere la seconda fase
- `complete`: non ha effetti sulla transazione

Perdita di messaggi

- la perdita di un messaggio `prepare` o del successivo `ready` non sono distinguibili dal TM
  - in entrambi casi, il timeout scade e una decisione globale di abortire viene presa
- la perdita di un messaggio di decisione (`commit`/`abort`) o di un `ack` non sono distinguibili
  - in entrambi i casi il timeout della seconda fase scade e la seconda fase è ripetuta

e **PARTIZIONAMENTO** della rete: TM e RM devono appartenere alla stessa partizione.

UNDO(disfare): in caso di fallimento della transazione deve essere possibile "disfare" l'azione svolta sui dati

REDO(rifare): se la transazione ha avuto successo ma le modifiche al DB non sono state rese permanenti, le modifiche vanno ripetute.

Domanda 1)
Elencare e descrivere in modo completo le *proprietà ACIDe* delle transazioni.
Indicare quali di queste proprietà cambiano a seguito della *distribuzione* della base di dati, fornendo la motivazione e un esempio adeguato.

ACID Atomicity, Consistency, Isolation, e Durability

- **ATOMICITÀ**: la transazione è **indivisibile** nella sua **esecuzione**: totale o nulla, non è parziale;
- **CONSISTENZA**: Lo stato iniziale del database: **coerente deve rimanere** tale anche quando la transazione termina ovvero non deve violare eventuali Vincoli di integrità
- **ISOLAMENTO**: ogni transazione deve essere eseguita in modo isolato e **indipendente** dalle altre transazioni, l'eventuale fallimento di una transazione non deve interferire con le altre in esecuzione;
- **DURABILITÀ**: **persistenza**, una volta che una transazione abbia richiesto un ***commit work***, i cambiamenti apportati non dovranno essere più persi. Per evitare che nel lasso di tempo fra il momento in cui la base di dati si impegna a scrivere le modifiche e quello in cui li scrive effettivamente si verifichino perdite di dati dovuti a malfunzionamenti, VENGONO TENUTI DEI REGISTRI DI LOG dove sono annotate tutte le operazioni sul DB.

La distribuzione dei dati non influenza
- consistenza: i vincoli di integrità descrivono solo proprietà locali
  - è un limite della corrente tecnologia DBMS
- persistenza: ogni sistema garantisce persistenza ai dati localmente memorizzati
  - meccanismi di recovery (log, checkpoint, dump) locali

La distribuzione dei dati influenza
- isolamento
- atomicità

Atomicità

Atomicità di transazioni distribuite può essere compromessa da guasti/malfunzionamenti
- guasti a nodi (software/hardware)
- perdite di messaggi: lasciano l'esecuzione di un protocollo in uno stato non certo
  - ogni messaggio del protocollo (*msg*) è seguito da un messaggio di conferma di ricezione (*ack*)
  - la perdita di uno dei messaggi lascia il mittente insicuro circa la sua ricezione
- guasti a link di comunicazione: possono causare partizionamenti della rete
  - una transazione può essere simultaneamente attiva in più di una sotto-rete

ISOLAMENTO: Controllo della concorrenza, SERIALIZZABILITA' GLOBALE **non solo locale**, garantita l'isolamento delle transazioni da 2PL/ TS

Domanda 1)
Nell'ambito del modello relazionale, dire cosa si intende per *vincolo di integrità referenziale*.
Indicare le operazioni che possono violarlo e spiegare quali controlli vengono eseguiti dal DBMS per verificarne la violazione e per assicurarne il soddisfacimento. Infine, elencare e descrivere le politiche di reazione che possono essere associate al vincolo di integrità referenziale in SQL.

DEFINIZIONE: Nell'ambito dei RDBMS, l'**integrità referenziale** è un vincolo di integrità di tipo **inter**relazionale ovvero una proprietà dei dati la quale, per essere soddisfatta, richiede che ogni valore di un attributo (colonna) di una relazione (tabella) esista come valore di un altro attributo in un'altra relazione.

CONTROLLO: Per verificarne la violazione bisogna controllare se non c'è una corrispondenza tra F.K. e P.K.

ESEMPIO cancellare un record che contiene un valore a cui fa riferimento una foreign key di un'altra tabella violerebbe l'integrità relazionale.

POLITICHE DI REAZIONE:

- CASCADE: "Propagazione/aggiornamento" dei valori della F.K. del corrispettivi valori della P.K.
- SET NULL: Il valore della F.K. viene settato a NULL
- SET **DEFAULT**: F.K. settato a un **valore di default** che viene aggiunto anche alla **P.K.** **come attributo con valori nulli**
- NO ACTION: Nessuna variazione F.K. MESSAGGIO DI ERRORE: Violazione

Domanda 1)

Con riferimento alla *proprietà di distributività degli operatori algebrici* rispetto a ∪ e −.

1a) Dire quali delle affermazioni sotto valgono, fornendo la *formula di equivalenza* nel caso valgano, o la *motivazione* nel caso non valgano, ed in ogni caso un esempio (per ciascun punto) che ne mostri la valenza o la non valenza.

- Distributività della selezione rispetto a ∪
- Distributività della selezione rispetto a −
- Distributività della proiezione rispetto a ∪
- Distributività della proiezione rispetto a −

1b) Per le situazione in cui *non vale* l'equivalenza, dire se possono esistere (e quali) condizioni sugli schemi degli operandi per cui l'equivalenza possa valere.

1c) Per le situazione in cui *vale* l'equivalenza, dire se possono esistere (e quali) condizioni sugli schemi degli operandi per cui l'equivalenza possa non valere.

STUDENTI(C.F.,Cognome,Nome)
IMP(C.F.,Cognome,Nome)

$\sigma_{Cognome='Rossi'}$ (STUDENTI ∪ IMP) ≡
$\sigma_{Cognome='Rossi'}$ (STUDENTI) ∪ $\sigma_{Cognome='Rossi'}$ (IMP)

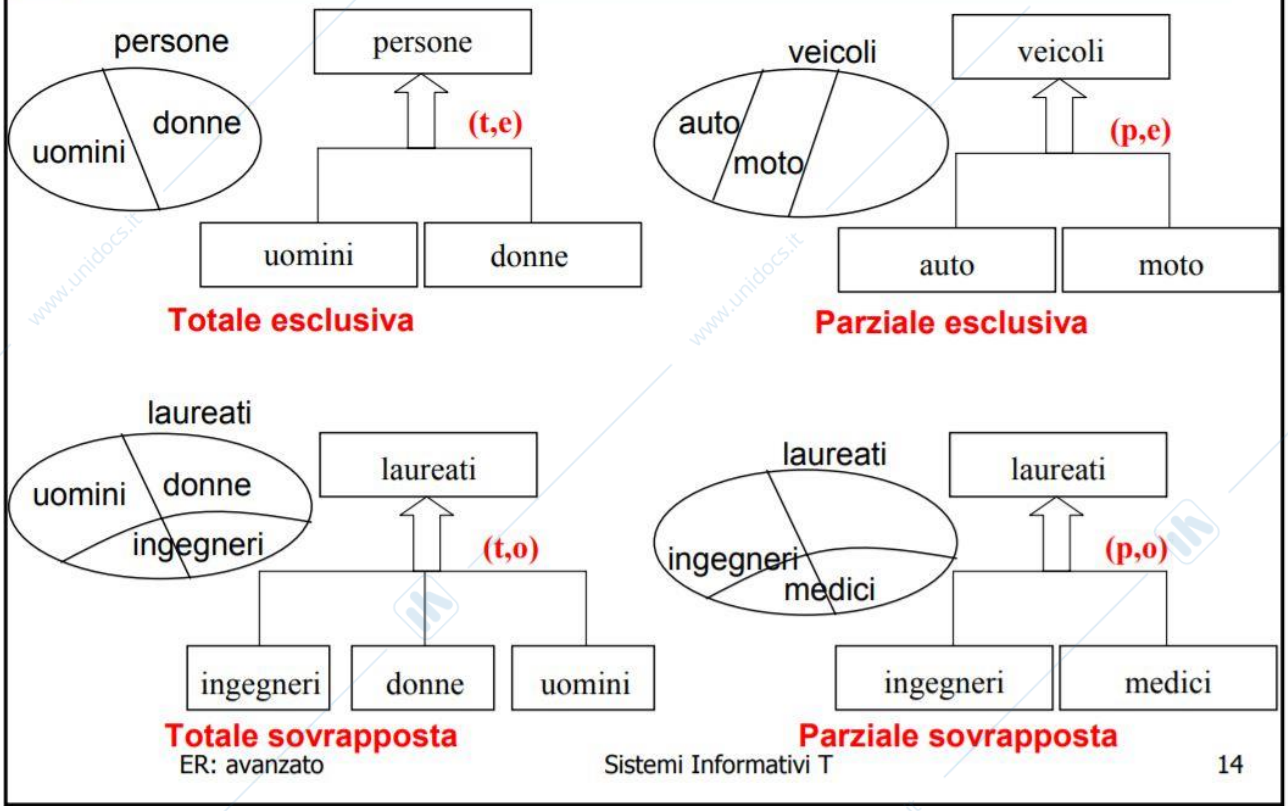
1a) - **selezione U**: $\sigma_C(E_1 \cup E_2) \equiv \sigma_C(E_1) \cup \sigma_C(E_2)$ – E_1 e E_2 espressioni; C condizione

- **Selezione -** identico sostituendo U con –
- **Proiezione** U $\prod_X(E_1 \cup E_2) \equiv \prod_X(E_1) \cup \prod_X(E_2)$ – E_1 e E_2 espressioni; X attributi in E_1 e E_2 X attributi, e non C condizioni
- **Proiezione -** **NON ESISTE**

1b) nella selezione quando

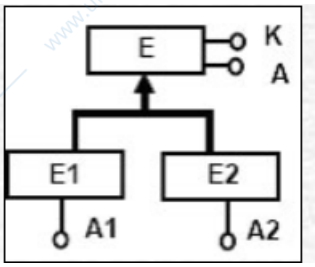
Domanda 1)
Nell'ambito del modello ER, illustrare le proprietà che caratterizzano le gerarchie di generalizzazione/specializzazione e quindi i diversi tipi di gerarchia che possono esistere. Per ciascuno di questi fornire degli esempi. Discutere inoltre i possibili approcci per l'eliminazione delle gerarchie nella ristrutturazione dello schema ER.

Proprietà di copertura - esempi

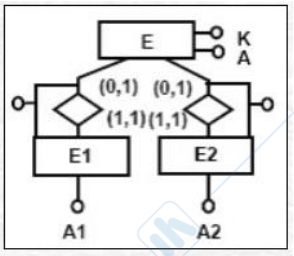


ER: avanzato Sistemi Informativi T 14

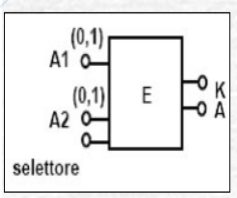
GERARCHIE



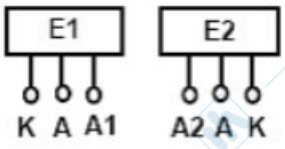
Mantenimento entita
Con associazioni



collasso verso L'ALTO(0,1)



collasso verso il BASSO



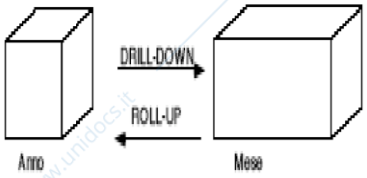
Domanda 1)
Caratterizzare in modo preciso, completo ed esauriente il modello multidimensionale dei dati e fornirne un esempio. Descrivere quindi le operazioni di *slice-and-dice*, *roll-up* e *drill-down* fornendo per ciascuna di queste un esempio.

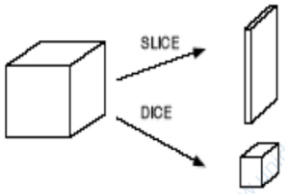
MODELLO MULTIDIMENSIONALE

è la rappresentazione dei dati ad **alto LV** , prescinde dai criteri di memorizzazione e favorisce l'analisi. Basato sul:

- FATTO: **concetto** sul quale fare l'analisi
- MISURA: **proprietà** atomica di un fatto da analizzare
- DIMENSIONE: descrive una **prospettiva** lungo la quale effettuare l'analisi

OPERAZIONI:



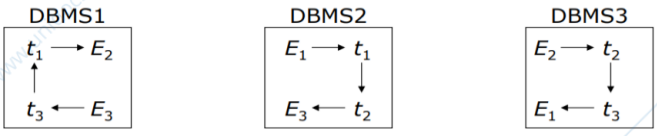


Catena di negozi
- fatto: vendita
- misure: quantità venduta, incasso
- dimensioni: articolo, tempo, luogo

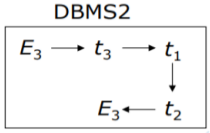
- **Slice-and-dice**: (seleziona e proietta) seleziona un **sottoinsieme** delle celle di un cubo
- **Roll-up**: aggrega i dati – applica una funzione aggregativa (tipicamente somma) sui dati aggregati di un cubo
- **Drill-down**: disaggrega i dati – è l'operazione inversa del roll-up – aggiunge dettaglio ad un cubo disaggregandolo lungo una o più dimensioni

Domanda 2)
Discutere il problema del *deadlock* in un *sistema distribuito* illustrando un possibile esempio di approccio per la sua rilevazione unitamente ad un esempio di esecuzione.

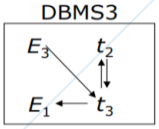
DEADLOCK: Situazioni *circolari di attesa* tra due o più nodi;
RILEVAZIONE: analizzando il grafo risultante per rilevare deadlock (periodicamente) E comunico le sequenze di attesa ad altri DBMS con un protocollo asincrono e distribuito (chiamate rpc).



DBMS1 comunica al DBMS2 ($E_3 \rightarrow t_3 \rightarrow t_1 \rightarrow E_2$)



DBMS2 comunica al DBMS3 ($E_3 \rightarrow t_3 \rightarrow t_2 \rightarrow E_3$)



deadlock!

Domanda 2)
Nell'ambito delle basi di dati distribuite:

2a) Descrivere la frammentazione e l'allocazione, descrivendo le proprietà che devono soddisfare.

2b) Descrivere i livelli di trasparenza che derivano dalla distinzione tra frammentazione ed allocazione, con un piccolo semplice esempio di query che faccia vedere la differenza dei livelli di trasparenza.

FRAMMENTAZIONE: suddividere una relazione in più frammenti
ORIZZONTALE (interpretato come selezione) e VERTICALE (proiezione)
Proprietà:

- ➔ *Completezza:* ogni dato di R deve essere **presente** in un qualche suo frammento R
- ➔ *Ricostruibilità:* R deve essere interamente **ricostruibile** a partire dai suoi frammenti

ALLOCAZIONE ogni frammento corrisponde a un file a livello fisico ed è allocato su uno specifico server
NON-RIDONDANTE: ogni frammento o relazione è allocato su 1 server
ridondante: *almeno* un frammento o relazione è allocato su più di un server

LIVELLI DI TRASPARENZA:
Prog. Non ha bisogno di conoscere l'allocazione/frammentazione

Frammentazione

```
procedure Query1(:fnum,:nome);
     select Nome into :nome
     from Fornitore
     where Fnum = :fnum;
end procedure;
```

Allocazione (Programmatore deve conoscere struttura frammenti)

```
procedure Query2(:fnum,:nome);
     select Nome into :nome
     from Fornitore1
     where Fnum = :fnum;
if :empty then
     select Nome into :nome
     from Fornitore2
     where Fnum = :fnum;
end procedure;
```

Domanda 2)
Rispondere brevemente, ma in modo completo, alle seguenti domande.

- Illustrare e spiegare brevemente il controllo eseguito dallo scheduler per decidere se accordare o meno una operazione per un controllo di concorrenza basato su *timestamp*. Illustrare sia il caso *monoversione* sia il caso *multiversione*.
- Nell'ambito del linguaggio di interrogazione SQL, dire se esistono *viste non modificabili*, spiegando il perché e fornendo, se esiste, un esempio.

MONOVERSIONE READ(X, TS)

TS>=WTM(X) RTM(X) = **max**(RTM(x), TS) else ABORT(x)

WRITE(X, TS)

TS>=RTM(X) and TS>=WTM(X) WTM(X)=TS else ABORT(x)

MULTIVERSIONE

read(x, ts)
- sempre accettata
- leggi la versione x_k tale che
  - $WTM(x_k) = max\{WTM(x_i) \mid WTM(x_i) \leq ts\}$
- $RTM(x_k) := max(RTM(x_k), ts)$

write(x, ts)
- trova la versione x_j tale che
  - $WTM(x_j) = max\{WTM(x_i) \mid WTM(x_i) \leq ts\}$
- $ts < RTM(x_j)$
  - richiesta è rifiutata, transazione abortita
- altrimenti
  - richiesta è accettata; crea nuova versione x_k
  - $RTM(x_k) := ts$; $WTM(x_k) := ts$

Domanda 2)
Rispondere brevemente, ma in modo preciso e completo, alle seguenti domande.

1. Nell'ambito della gestione delle transazioni, elencare e descrivere i *livelli di isolamento* previsti da SQL.
2. Nell'ambito delle basi di dati attive, descrivere i componenti dei trigger insieme con i *livelli di granularità* e le *modalità di attivazione*.
3. Nell'ambito del controllo dell'accesso in SQL, indicare il significato e l'utilizzo di `references`.

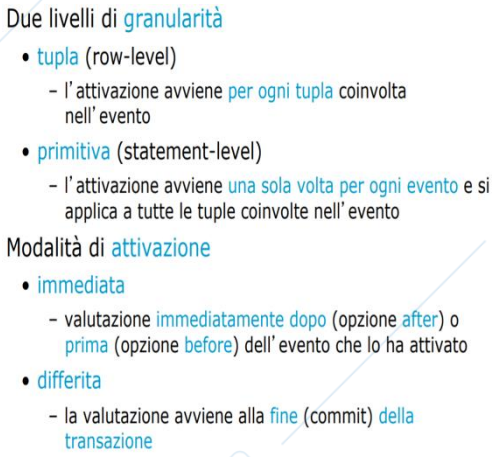
LIVELLI DI ISOLAMENTO SQL(-92)

- **READ UNCOMMITTED:** Non sono isolate, tipicamente solo LETTURA
- **READ COMMITTED:** La transazione resta in attesa fino a quando non vengono sbloccate le righe con blocco Write da altre transazioni; si impedisce la lettura di dati "Dirty" (lettura dati di cui non è stato ancora effettuato il COMMIT)
- **REPEATABLE READ:** previene le read dirty e La transazione **include i blocchi di lettura** in tutte le righe restituite all'applicazione e scrive i blocchi in tutte le righe inserite, aggiornate o eliminate
- **SERIALIZABLE:** previene le read dirty , La transazione include un blocco di lettura (se legge solo le righe) o il **blocco di scrittura** (se può aggiornare o eliminare righe) **nell'intervallo di righe a cui ha effetto**

2) COMPONENTI **TRIGGER:**

"Ad un particolare evento che verifica una condizione specifica esegui un'azione specifica"

- **EVENTO:** Utilizzo le primitive di sql per la **manipolazione dati** es: insert, delete, update
- **CONDIZIONE**: Predicato **bool**eano espresso in sql
- AZIONE: sequenze di primitive sql/procedure



REFERENCES è una parola che chiave per definire il vincolo di integrità referenziale F.K con la P.K.

```
FOREIGN KEY (Studente) REFERENCES Alunni(Codice)
```

ALUNNI (tabella) CODICE(P.K.)

Domanda 2)
Con riferimento al data warehousing, illustrare il significato di una query della forma

```
select D1.L1,..., Dn.Ln,Aggr1(F.M1),..., Aggrk(F.Mk)
from   Fatti as F, Dimens1 as D1,..., Dimensn as Dn
where Join-predicate(F,D1) ...
      and Join-predicate(F,Dn)
      and selection-predicate
group by D1.L1,..., Dn.Ln
order by D1.L1,..., Dn.Ln
```

illustrare poi una specifica instanziazione di uno schema a stella e di relativa query della forma sopra, spiegandone la semantica.

- `F.Mi`: misura *i*-esima della tabella dei fatti `F`
- `Di.Li`: livello della *i*-esima tabella dimensione
- `Aggri`: funzione aggregativa
- `Join-predicate(F,Di)` predicato di join fra `Fatti` e `Di`
- `selection-predicate`: condizione di selezione sulle tabelle dimensione

```
select P.Categoria, T.Trimestre, sum(V.Unità)
from Vendite as V, Prodotto as P, Tempo as T
where V.CodiceProdotto = P.CodiceProdotto
      and V.CodiceTempo = T.CodiceTempo
      and T.Anno = 2003
group by P.Categoria, T.Trimestre
order by P.Categoria, T.Trimestre
```

Seleziona le vendite complessive del 2003 per categoria di articolo e trimestre

Domanda 2)
Nell'ambito della gestione della concorrenza basata su *timestamp*, spiegando sia il caso *monoversione* sia il caso *multiversione*:

2a) definire cosa sono i timestamp e come vengono assegnati alle transazioni;

2b) spiegare che informazione viene mantenuta in associazione alle risorse;

2c) riportare precisamente e formalmente le informazioni mantenute e i controlli necessari per verificare, per ciascuna operazione, se può essere accettata o deve essere rifiutata e l'aggiornamento delle informazioni associate alle risorse;

2d) illustrare inoltre, se esiste, un esempio di schedule composto dal minimo numero di operazioni:

- TS monoversione ma non TS multiversione;
- TS multiversione ma non TS monoversione;

nel caso si pensi che tale schedule non esista, darne la ragione.

2a) Il Timestamp è un valore asssegnato a ciascuna transazione per identificarne l'inizio temporale. Viene assegnato in ordine "crescente", dunque, una transazione avrà TS >= TS che lo ha preceduto.

2d) TS MONO E NON MULTI

Domanda 3)
Descrivere le condizioni che un documento XML deve soddisfare per essere:

1. ben formato
2. valido

Documento ben formato

DTD e XML Schema

Permettono di definire un modello per i documenti

- dettano il tipo dei documenti
  - i tag (elementi) ammessi e loro proprietà
    - cardinalità, attributi (dominio e valori di default), ...
  - le regole di annidamento dei tag

Un documento che è validato rispetto a un DTD/XML schema è detto valido (valid)

Un documento è detto ben formato (well formed) se

- inizia con il prologo
  - es., `<?xml version="1.0"?>`
- tutti gli elementi hanno tag iniziali e finali
- la nidificazione dei tag è corretta
- gli attributi sono correttamente codificati
  - valori tra apici
- i valori sono correttamente codificati
  - case sensitive

Domanda 3)
Sia T un insieme di n transazioni ciascuna delle quali composta da una singola operazione di scrittura su uno stesso oggetto. Sia S uno schedule di tali transazioni. Indicare, se possibile, numero minimo e massimo di schedule seriali rispetto a cui S è:

- view-equivalente
- conflict-equivalente

Si assuma ora che una delle transazioni faccia una lettura (invece della scrittura). Indicare, se possibile, numero minimo e massimo di schedule seriali rispetto a cui S è:

- view-equivalente
- conflict-equivalente

BUUUUUUUUU

Domanda 3)

1. Si consideri la tecnica di *prevenzione dei deadlock basata sui timestamp*.
   (a) Dire quale transazione viene uccisa nel caso:
   - preemptive
   - non preemptive

   indicando anche il controllo fatto per decidere se uccidere o mettere in attesa una risorsa.
   (b) Quale timestamp viene assegnato alla successiva attivazione della transazione abortita? Perché?
2. Nell'ambito delle *basi di dati distribuite* illustrare un approccio per la rilevazione e risoluzione di deadlock.

ti richiede lock su x
tj ha lock su x

PREEMPTIVE: (interrompenti) uccidono la transazione che possiede la risorsa
ts(ti) > ts(tj): ti aspetta altrimenti abort(tj)

NON PREEMPTIVE (Non interrompenti) uccidono la transazione che richiede la risorsa
ts(ti) < ts(tj): ti aspetta altrimenti abort(ti)

b) viene assegnato lo stesso TS della transazione abortita altrimenti potrebbero venire sempre uccise causando il problema della **STARVATION**.

2) **2 Phase Commit** basi di dati distribuite / **GRAFO ATTESE** (link **bi**-direzionale tra transazioni)

Domanda 3)
Rispondere in modo preciso e completo alle seguenti domande.

1. Descrivere le regole *Write Ahead Log* e *Commit-Precedenza* e dire perché servono.
2. Nell'ambito del linguaggio di interrogazione SQL, dire se esistono *viste non modificabili*, spiegando il perché e fornendo, se esiste, un esempio.

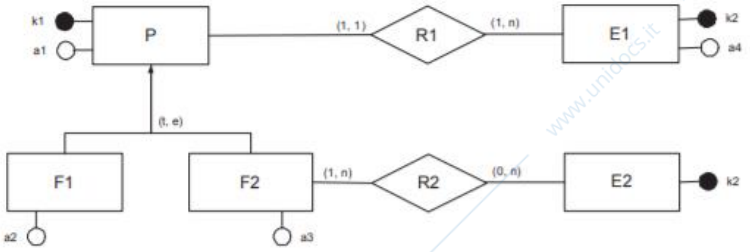
WRITE AHEAD LOG: fornire atomicità e durata .
Le modifiche vengono prima registrate nel registro, poi scritte nel database.
consente di eseguire gli aggiornamenti di un database sul posto" .
Il vantaggio principale di eseguire aggiornamenti sul posto è che riduce la necessità di modificare indici e bloccare elenchi.

COMMIT PRECEDENZA: i record di log **siano scritti PRIMA** dell'**esecuzione** dell'operazione di commit (regola di commit-precedenza).

Si esistono perché non è sempre possibile determinare un modo univoco in cui la modifica sulla vista possa essere riportata sulle tabelle di base; infatti si incontrano grossi problemi quando la vista è definita tramite join di due tabelle. Lo standard SQL permette che una vista sia aggiornabile solo quando le sue righe corrispondono alla tabella di base. Per modificare solo la vista, senza intaccare le tabelle di base, si usa il comando check option. Nel caso di viste ottenute da altre viste, esistono le due opzioni [local | cascade]: local effettua un controllo solo sull'ultimo livello delle viste mentre cascade effettua un controllo su tutti i livelli di definizione delle viste.

Domanda 3)
Illustrare i concetti di *regole di vincolo* e di *derivazione* e riportare le regole di vincolo e di derivazione relative alla progettazione dell'Esercizio 4.

Domanda 4)
Sia dato il seguente schema ER:



Disegnare, utilizzando gli appositi spazi nel foglio allegato, la traduzione dello schema applicando le tre politiche di eliminazione della gerarchia ISA.
Si ricorda di indicare, negli schemi risultanti **tutte** le *relazioni*, *entità*, *attributi*, e *cardinalità* che ne fanno parte. Elementi mancanti saranno considerati come errati.

Domanda 3)
Dopo aver spiegato i concetti di *terminazione*, *confluenza* e *determinismo* delle osservazioni, dire se il seguente insieme di trigger è terminante oppure no, motivando opportunamente la risposta.

Trigger 1

```
CREATE TRIGGER CalcolaSaldi
AFTER UPDATE OF Prezzo ON Prodotto
REFERENCING new_table AS NuovoPrezzo
BEGIN
  UPDATE Prodotto
  SET Sconto = Prezzo * 0.20
  WHERE IdProdotto IN (
    SELECT IdProdotto
    FROM NuovoPrezzo );
END;
```

Trigger 2

```
CREATE TRIGGER VerificaSconto
AFTER UPDATE OF Sconto ON Prodotto
REFERENCING new_table as NuovoProd
WHEN 10 > (
  SELECT Sconto
  FROM Prodotto
  WHERE Id IN
    (SELECT Id FROM NuovoProd))
BEGIN
  UPDATE Prodotto
  SET Prezzo = 0.90 * Prezzo;
END;
```

TERMINAZIONE: produce stato finale

CONFLUENZA: rispetta l'ordine

DETERMINISMO DELLE OSSERVAZIONI: confluenza e producono la stessa sequenza di azioni visibili

DBMS sono progettati per **OLTP** (strutture di memorizzazioni, indici, **transazioni: " CHE COSA STA SUCCEDENDO?"**)

DW sono progettati per **OLAP:** interrogazioni complesse con funzioni statistiche, visti multidimensionali, dati storici: **"CHE COSA SUCCEDERA'?"**

	OLTP	OLAP
Finalità	Supporto all'operatività	Supporto al processo decisionale
Utenti	Molti, livello operativo	Pochi, livello direzionale
Modalità di utilizzo	Guidata, per processi e stadi successivi	Interrogazione ad hoc
Quantità di dati per operazione elementare	Bassa: centinaia di record per ogni transazione	Alta: milioni di record per ogr query
Qualità	In termini di integrità	In termini di consistenza
Orientamento	Per processo/applicazione	Per soggetto
Frequenza di aggiornamento	Continua, tramite azioni	Sporadica, tramite funzioni esplicite
Copertura temporale	Dati correnti	Storica
Ottimizzazione	Per accessi in lettura e scrittura su una porzione di dati	Per accessi in sola lettura su tutta la base di dati

Dal modello concettuale al modello logico: le regole di derivazione

- ogni *entità* diventa una tabella
- ogni *attributo* di un'entità diventa un attributo della relazione (nome di una colonna della tabella)
- ogni colonna della relazione eredita le caratteristiche dell'attributo dell'entità da cui deriva
- l'identificatore univoco di un'entità diventa la *chiave primaria* della relazione derivata

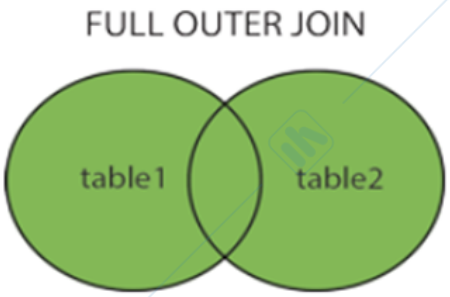
Associazioni

- l'associazione *uno a uno* diventa un'unica relazione che contiene gli attributi della prima e della seconda entità
- l'identificatore univoco dell'entità di partenza nell'associazione *uno a molti* diventa **chiave esterna** (*foreign key*) dell'entità di arrivo associata, cioè i suoi attributi - identificatori univoci - diventano colonne della seconda relazione
- l'associazione *molti a molti* diventa una nuova relazione (in aggiunta alle relazioni derivate dalle entità) composta dagli identificatori univoci delle due entità e dagli eventuali attributi dell'associazione.

Domanda 2)
Descrivere il problema legato alla presenza dei valori nulli nel modello relazionale. Spiegare come il linguaggio SQL gestisce tali valori e come le tabelle di verità degli operatori logici (`AND`, `OR` e `NOT`) vengono estese per tenere conto anche di questo tipo di valore.

Domanda 2)
Con riferimento a SQL, spiegare la differenza fra: *inner join*, *left-outer join* e *right-outer join*, spiegando perché ciascuno di essi serve ed illustrandone un esempio.

```
SELECT column_name(s)
FROM table1
LEFT/OUTER/FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```



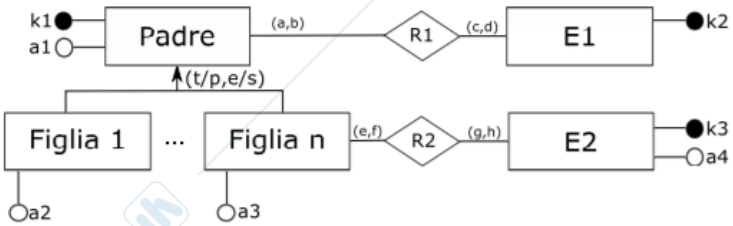
Domanda 1)
Illustrare e commentare le diverse *fasi del ciclo di vita* di una base di dati, specificando cosa ogni singola fase prende in ingresso e cosa produce. Nella descrizione, discutere in modo più approfondito i diversi passi della progettazione.

1. **Studio di fattibilità:** si prendono in considerazione tutte le possibili implementazioni realizzabili, si calcolano i relativi costi implementativi e si danno priorità alle varie fasi di sviluppo;
2. **Raccolta e analisi dei dati:** si prendono in considerazione tutti i dati che dovranno essere gestiti e come e quali utenti dovranno accedere agli stessi; in questa fase si produce anche uno schema astratto della base di dati. Vengono inoltre stabiliti i requisiti HW e SW del sistema informativo.
3. **Progettazione:** in prima analisi dei dati, ovvero si progetta la componente intensionale della base di dati; in seconda analisi si progettano i programmi applicativi che andranno ad interagire con i dati.
4. **Implementazione:** viene creato il database su un server di Database e viene scritto il codice dei programmi applicativi.
5. **Test e validazione:** viene testato il funzionamento del sistema informativo ed in caso di bug vengono subito cercate delle soluzioni.
6. **Funzionamento:** il sistema informativo viene utilizzato per il suo scopo originale ed in caso di corretto funzionamento necessita solo di azioni di gestione e mantenimento.

PROGETTAZIONE:

-CONCETTUALE (E-R, formalizzazione idee)
– **LOGICA** (Ristrutturazione E-R: eliminazione gerarchie in alto/basso, attributi multivalori, traduzione nel modello relazionale)
– **FISICA** (tabelle, relative ad un DB)

Domanda 1)
Sia data una generarchia di generalizzazione, caratterizzata da un'entità PADRE e da *n* entità FIGLIE.



1. Spiegare i diversi approcci per l'*eleminazione delle gerarchie* nella *ristrutturazione* dello schema ER.
2. Disegnare, utilizzando gli appositi spazi nel foglio allegato, la traduzione dello schema sopra riportato applicando le tre politiche di eliminazione della gerarchia ISA (aggiungendo eventuali entità, relazioni e/o attributi, se necessario), indicando anche l'eventuale informazione persa nello schema e quindi eventuali regole di vincolo e restrizioni dei valori di dominio che devono essere specificati.
3. Per ciasuno dei tre possibili tipi di ristrutturazione (collasso verso l'alto, collasso verso il basso, mantenimento delle entità), si compili la tabella allegata indicando:
   - il numero di entità
   - il numero di relazioni
   - il numero di attributi selettori necessari
   - la cardinalità del dominio dei selettori necessari

   che caratterizzano il risultato dell'eleiminazione della gerarchia (senza considerare le relazioni $R1$, $R2$ e le entità $E1$ ed $E2$).

Rispondere alla domanda considerando ciascun tipo di gerarchia di generalizzazione (ovvero (t,e), (t,s), (p,e) e (p,s)).

FUNZIONI CONDIZIONALI SQL

COALESCE

trova il CdL di ogni studente, utilizzando la stringa "non ins." nel caso non sia inserito

```
select Matr,Cognome,Nome,coalesce(CdL,'non ins.')
from Studenti
```

NULLIF: Restituisce NULL se i due valori **COINCIDONO** , altrimenti il valore dell'espressione altrimenti

trova gli esami, utilizzando il valore nullo nel caso il voto sia "ritirato"

```
select Matr,Cod-corso,Data,nullif(Voto,'ritirato')
from Esami
```

Domanda 1)

1a) Nell'ambito del modello ER, illustrare le proprietà che caratterizzano le gerarchie di generalizzazione/specializzazione e quindi i diversi tipi di gerarchia che possono esistere. Per ciascuno di questi fornire degli esempi.

1b) Discutere i possibili approcci per l'eliminazione delle gerarchie nella ristrutturazione dello schema ER.

1c) Per ciascun approccio fornire inoltre l'esempio, eliminando la gerarchia in figura, compilando il foglio allegato riportando:

- lo schema ristrutturato;
- l'eventuale informazione persa nella ristrutturazione e che dovrà quindi essere rappresentata come regola.