INFORMATICA DOCUMENTALE

Laurea in Scienze della Comunicazione – Università degli Studi di Verona

VERIFICA SCRITTA DEL 13 FEBBRAIO 2014

matricola	cognome	nome	firma

A.1 + A.2 + A3	B.1	B.2	B.3	Totale

Istruzioni

- È vietato portare all'esame libri, eserciziari, appunti e dispense. Chiunque venga trovato in possesso di documentazione relativa al corso – anche se non attinente alle domande proposte – vedrà annullata la propria prova.
- Scrivere solo sui fogli distribuiti, cancellando le parti di brutta con un tratto di penna. Non separare questi fogli.
- Tempo a disposizione: 1 ora e 45 minuti.

A. Parte prima

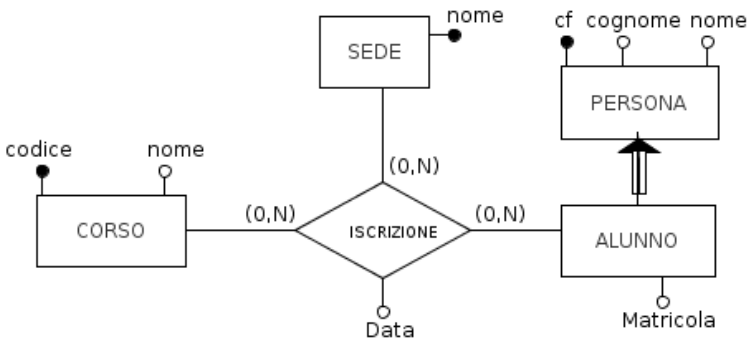
A.1. In al più 10 righe descrivere quali vincoli esprimono le cardinalità nel modello E/R.

A.2. In al più 5 righe spiegare il concetto di atomicità di una transazione e fornirne un semplice esempio.

A.3. In al più 5 righe descrivere che cos'è un DBMS.

B. Parte seconda

B.1. Considerare il seguente schema ER.



B.1.a. Spiegare cosa significa (0,N) vicino all'entità CORSO.

B.1.b. Descrivere l'associazione ISCRIZIONE, discutendo anche le cardinalità.

B.1.c. Descrivere il tipo di legame tra PERSONA e ALUNNO.

B.1.d. Sarebbe possibile imporre il vincolo che certi corsi siano svolti solo in determinate sedi?

B.1.e. Tradurre lo schema ER in uno schema relazionale, indicando in quest'ultimo eventuali chiavi, vincoli di non nullità e vincoli di integrità referenziale.

B.2. Considerare il seguente schema relazionale che rappresenta una parte di un sistema per la gestione di una "Banca del tempo", costituita da persone che si associano per scambiare servizi e/o saperi, attuando un aiuto reciproco:

`SOCIO(codice, cognome, nome, indirizzo, telefono*)`

`CATEGORIA(id, descrizione)`

`OFFERTA(socio, categoria)`

`PRESTAZIONE(numero, fruitore, erogatore, categoria, data, durata)`

con vincoli di integrità referenziale tra l'attributo `socio` di `OFFERTA` e l'attributo `codice` di `SOCIO`, tra l'attributo `erogatore` di `PRESTAZIONE` e l'attributo `codice` di `SOCIO`, tra l'attributo `fruitore` di `PRESTAZIONE` e l'attributo `codice` di `SOCIO`, tra l'attributo `categoria` di `PRESTAZIONE` e l'attributo `id` di `CATEGORIA`, tra l'attributo `categoria` di `OFFERTA` e l'attributo `id` di `CATEGORIA` e con il vincolo che in `SOCIO` `fruitore` e `erogatore` siano necessariamente diversi in una stessa prestazione e la seguente istanza:

SOCIO				
codice	**cognome**	**nome**	**indirizzo**	**telefono**
001	Neri	Laura	laura.neri@opera.it	NULL
002	Bianchi	Gianni	gianni.bianchi@opera.it	NULL
003	Verdi	Giuseppe	giuseppe.verdi@opera.it	340 11223344

CATEGORIA	
id	**descrizione**
1	Riparazione idraulica
2	Riparazione falegnameria
3	Riparazione elettrica
4	Riparazione informatica

OFFERTA	
socio	**categoria**
001	1
001	2
002	1
002	2
002	4
003	3

PRESTAZIONE					
numero	**fruitore**	**erogatore**	**categoria**	**data**	**durata**
1	002	001	1	2013-12-13	2.50
2	002	001	2	2013-12-13	3.00
3	003	001	2	2013-12-17	1.00
4	001	002	1	2013-11-13	2.00
5	003	002	2	2013-12-09	1.50
6	002	003	3	2013-12-22	4.00

B.2.a. Scrivere i comandi SQL per creare le quattro tabelle sopra riportate (incluse eventuali chiavi, vincoli di non nullità, vincoli di unicità e vincoli di integrità referenziale). Tenere conto che deve essere evitato di cancellare record, da cui dipendono altri record. Le modifiche di chiavi primarie, invece, si "propagano" nel database alle chiavi secondarie.

B.2.b. Se non lo si è già previsto come vincolo al punto precedente, come posso modificare lo schema SQL della tabella `PRESTAZIONE` in modo tale che ogni prestazione sia effettivamente associata ad un erogatore che offre il tipo di prestazione erogato?

B.2.c. Cosa restituisce la seguente interrogazione? (scrivere la tabella risultante)

```
SELECT cognome FROM socio WHERE cognome LIKE '%e%' ORDER BY cognome;
```

B.2.d. Cosa restituisce la seguente interrogazione? (scrivere la tabella risultante)

```
SELECT S.cognome, SUM(P.durata) AS ore_prestate
FROM socio AS S INNER JOIN prestazione AS P ON S.codice = P.erogatore
GROUP BY S.cognome HAVING SUM(P.durata) >= 4;
```

B.2.e. Scrivere i comandi SQL per cancellare dal database il socio "Laura Neri".

B.3. Si consideri il seguente schema SQL, che rappresenta un semplice sistema di gestione di un catalogo di videogiochi.

```
CREATE TABLE produttore (
    nome VARCHAR(40) PRIMARY KEY,
    sito_web VARCHAR(250) NOT NULL
);
CREATE TABLE tipologia (
    id INT PRIMARY KEY,
    descrizione VARCHAR(50)
);
CREATE TABLE videogioco (
    codice INT PRIMARY KEY,
    titolo VARCHAR(100),
    produttore VARCHAR(40) NOT NULL,
    tipologia INT NOT NULL,
    descrizione VARCHAR(250),
    FOREIGN KEY(produttore) REFERENCES produttore(nome) ON DELETE CASCADE ON UPDATE
CASCADE,
    FOREIGN KEY(tipologia) REFERENCES tipologia(id) ON DELETE CASCADE ON UPDATE
CASCADE
);
CREATE TABLE personaggio (
    codice INT PRIMARY KEY,
    nome VARCHAR(30),
    videogioco INT,
    FOREIGN KEY(videogioco) REFERENCES videogioco(codice) ON DELETE CASCADE ON UPDATE
CASCADE
);
```

scrivere i comandi SQL che permettono di:

B.3.a. Ottenere la lista dei nomi dei produttori non ripetuti che hanno prodotto videogiochi in ordine descrescente.

B.3.b. Ottenere la lista dei videogiochi per i videogiochi che hanno più di 2 personaggi, con titolo, produttore e il relativo numero dei personaggi.

B.3.c. Ottenere la lista dei videogiochi con titolo, produttore e descrizione della tipologia ordinati per tipologia prodotti da 'Ubisoft' o 'Artematica'.

B.3.d. Aggiornare nell'intero database nome del produttore 'Ubisoft' a 'Ubisoft Italia'.

B.3.e. Rimuovere le tipologie, alle quali non è stato assegnato alcun videogioco.

SOLUZIONI

A. Parte prima

A.1. Le cardinalità vengono utilizzate per esprimere vincoli di partecipazione minima e massima di un'occorrenza di un'entità ad una relazione con un altra o altre entità oppure in riferimento ad un attributo per specificare il “grado” minimo e massimo di un attributo per singola occorrenza. Un esempio di quest'ultimo caso è quello di un attributo indirizzo di un'entità persona che può avere cardinalità (n,m), dove n indica il numero minimo di indirizzi, che può avere una persona: se n è 0 l'attributo è opzionale mentre se è 1 è obbligatorio, m indica il numero massimo di indirizzi. Normalmente la cardinalità (1,1) si omette.

A.2. Con atomicità di una transazione indichiamo la caratteristica per cui una sequenza di comandi con “esecuzione indivisibile"" su una base di dati: o viene eseguita del tutto o per nulla (non è ammessa l'esecuzione parziale). Come esempio, possiamo pensare ad una transazione bancaria con il passaggio di denaro da un conto corrente ad un altro. Perché la transazione sia consistente prelievo e versamento devono avvenire entrambi oppure nessuno dei due.

A.3. Un Database Management System è un sistema software, che gestisce i dati di un database (collezioni di dati grandi, condivise e persistenti) e regola gli accessi agli stessi da parte di utenti e applicazioni con opportune politiche di sicurezza/privatezza e cercando di garantire coerenza, efficacia ed efficienza.

B. Parte seconda

B.1.

B.1.a. Non è obbligatorio che ogni corso partecipi all'associazione ISCRIZIONE, il che significa che ci possono essere corsi senza iscrizioni.

B.1.b. ISCRIZIONE è un'associazione ternaria. Come già illustrato al punto precedente per l'entità CORSO, anche per SEDE e ALUNNO non è obbligatorio partecipare all'associazione, quindi ci possono essere sedi e alunni senza iscrizioni.

B.1.c. PERSONA è una generalizzazione di ALUNNO. Si tratta di una generalizzazione parziale in quanto non tutte le persone sono alunni.

B.1.d. È un vincolo interrelazionale che non si riesce ad esprimere nella notazione E-R. È eventualmente possibile aggiungere una nota che lo descriva.

B.1.e. `CORSO(codice, nome)`
`ALUNNO(cf, cognome, nome, matricola)`
`SEDE(nome)`
`ISCRIZIONE(corso, sede, alunno, data)`
con vincolo di integrità referenziale tra l'attributo `corso` dell'entità `ISCRIZIONE` e l'entità `CORSO`, tra l'attributo `sede` di `ISCRIZIONE` e l'entità `SEDE` e tra l'attributo `alunno` sempre di `ISCRIZIONE` e l'entità `ALUNNO`.

B.2.

B.2.a.

```
CREATE TABLE socio (
    codice CHAR(3) PRIMARY KEY,
    cognome VARCHAR(30) NOT NULL,
    nome VARCHAR(30) NOT NULL,
    indirizzo VARCHAR(60) NOT NULL,
    telefono VARCHAR(20)
);

CREATE TABLE categoria (
    id INT PRIMARY KEY,
    descrizione VARCHAR(250) NOT NULL
);

CREATE TABLE offerta (
    socio VARCHAR(3),
    categoria INT,
    PRIMARY KEY(socio, categoria),
    FOREIGN KEY(socio) REFERENCES socio(codice) ON DELETE NO ACTION ON UPDATE CASCADE,
    FOREIGN KEY(categoria) REFERENCES categoria(id) ON DELETE NO ACTION ON UPDATE CASCADE

);

CREATE TABLE prestazione (
    numero INT PRIMARY KEY,
    fruitore VARCHAR(3) NOT NULL,
    erogatore VARCHAR(3) NOT NULL,
    categoria INT NOT NULL,
    data DATE NOT NULL,
```

```
    durata DECIMAL(4,2) NOT NULL,
    FOREIGN KEY(fruitore) REFERENCES socio(codice) ON DELETE NO ACTION ON UPDATE CASCADE,
    FOREIGN KEY(erogatore) REFERENCES socio(codice) ON DELETE NO ACTION ON UPDATE
CASCADE,
    FOREIGN KEY(categoria) REFERENCES categoria(id) ON DELETE NO ACTION ON UPDATE
CASCADE,
    CHECK (fruitore <> erogatore)
);
```

B.2.b. Si usa il seguente vincolo:

```
FOREIGN KEY(erogatore, categoria) REFERENCES offerta(socio, categoria) ON DELETE
NO ACTION ON UPDATE CASCADE,
```

B.2.c.

cognome
Neri
Verdi

B.2.d.

cognome	ore_prestate
Neri	6.50
Verdi	4.00

B.2.e. In base ai vincoli precedenti, per eliminare un socio devo prima eliminare le sue eventuali "offerte" e "prestazioni".

```
DELETE FROM prestazione WHERE erogatore = (SELECT codice FROM socio WHERE cognome = 'Neri'
AND nome = 'Laura');
DELETE FROM prestazione WHERE fruitore = (SELECT codice FROM socio WHERE cognome = 'Neri'
AND nome = 'Laura');
DELETE FROM offerta WHERE socio = (SELECT codice FROM socio WHERE cognome = 'Neri' AND
nome = 'Laura');
DELETE FROM socio WHERE cognome = 'Neri' AND nome = 'Laura';;
```

B.3.

B.3.a. `SELECT DISTINCT produttore FROM videogioco ORDER BY produttore DESC;`

B.3.b.
```
SELECT titolo, produttore, COUNT(*) FROM videogioco INNER JOIN personaggio ON
videogioco.codice = personaggio.videogioco GROUP BY titolo, produttore HAVING COUNT(*) >
2;
```

B.3.c.
```
SELECT  titolo, produttore, tipologia.descrizione AS tipologia FROM videogioco INNER JOIN
tipologia ON videogioco.tipologia = tipologia.id WHERE produttore IN ('Ubisoft',
'Artematica') ORDER BY tipologia.descrizione;
```

B.3.d. `UPDATE produttore SET nome = 'Ubisoft Italia' WHERE nome = 'Ubisoft';`

B.3.e. `DELETE FROM tipologia WHERE id NOT IN (SELECT tipologia FROM videogioco);`