INFORMATICA DOCUMENTALE

Laurea in Scienze della Comunicazione – Università degli Studi di Verona

VERIFICA SCRITTA DEL 25 SETTEMBRE 2013

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

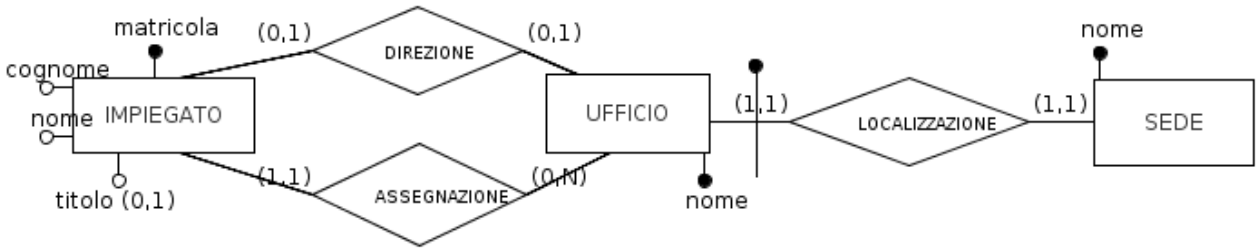
A.1. In al più 10 righe descrivere quali vincoli esprimono le cardinalità nel modello E/R.

A.2. In al più 5 righe descrivere la differenza nei sistemi informativi tra informazione e dato, Fornire un esempio al riguardo.

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 (0,1) vicino all'attributo titolo.

B.1.b. Descrivere la relazione che intercorre tra IMPIEGATO e UFFICIO, discutendo anche le cardinalità.

B.1.c. Descrivere l'identificatore dell'entità UFFICIO e in particolare dire se la cardinalità verso LOCALIZZAZIONE è necessario che sia (1,1) o potrebbe essere diversa.

B.1.d. Se si volesse rappresentare anche la data di assegnazione di un impiegato ad un ufficio come si potrebbe modificare l'ER?

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 :

```
ATLETA(nome, cognome, data_nascita*, squadra)
SQUADRA(nome, anno_fondazione*, citta)
PARTITA(squadra1, squadra2, data)
CITTA(nome)
```

con vincoli di integrità referenziale tra l'attributo squadra di ATLETA e l'attributo nome di SQUADRA, tra l'attributo citta di SQUADRA e l'attributo nome di CITTA e infine tra l'attributo squadra1 di PARTITA e l'attributo nome di SQUADRA e tra tra l'attributo squadra2 di PARTITA e l'attributo nome di SQUADRA, e la seguente istanza:

ATLETA			
nome	**cognome**	**data_nascita**	**squadra**
Bruno	Conti	NULL	Roma
Fulvio	Collovati	1957-05-09	Inter
Gabriele	Oriali	1952-11-25	Inter
Gaetano	Scirea	NULL	Juventus
Dino	Zoff	1942-02-28	Juventus
Antonio	Cabrini	1957-10-08	Juventus

SQUADRA		
nome	**anno_fondazione**	**citta**
Juventus	1897	Torino
Inter	NULL	Milano
Roma	1927	Roma

CITTA
nome
Milano
Torino
Roma

PARTITA		
squadra1	**squadra2**	**data**
Juventus	Inter	2013-03-12
Roma	Juventus	2013-03-19
Roma	Inter	NULL
Inter	Juventus	NULL

B.2.a. Scrivere i comandi SQL per creare le tre 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 una squadra coinvolta in qualche partita o nella quale milita un atleta. Stesso discorso per quanto riguarda l'eliminazione di una città. Le modifiche di chiavi primarie, invece, si "propagano" nel database alle chiavi secondarie.

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

```
SELECT cognome FROM atleta WHERE cognome LIKE 'C%' ORDER BY cognome;
```

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

```
SELECT S.nome, S.citta
FROM squadra AS S INNER JOIN partita AS P ON S.nome = P.squadra1
WHERE S.nome IN (SELECT squadra FROM atleta GROUP BY squadra HAVING COUNT(*) > 1);
```

B.2.d. Scrivere i comandi SQL per cancellare dal database la squadra Juventus con le relative partite.

B.2.e. Dire se i seguenti comandi sono corretti e in tal caso scrivere quanti record avranno dopo la loro esecuzione le tabelle PARTITA e SQUADRA sull'istanza sopra riportata, oppure se non lo è spiegare il motivo (max 3 righe).

```
UPDATE squadra SET nome = 'Roma AS' WHERE nome = 'Roma';
DELETE FROM partita WHERE data IS NULL AND squadra1 <> 'Roma AS';
```

B.3. Si consideri il seguente schema SQL, che rappresenta un semplice sistema di gestione ordini di articoli da parte di clienti.

```
CREATE TABLE cliente (
    nome VARCHAR(30) PRIMARY KEY,
    indirizzo VARCHAR(30) NOT NULL,
    citta VARCHAR(30) NOT NULL
);

CREATE TABLE articolo (
    codice VARCHAR(10) PRIMARY KEY,
    descrizione VARCHAR(30) NOT NULL,
    costo_unitario DECIMAL(10,2) -- Currency in euro with cents
);

CREATE TABLE ordine (
    numero INT PRIMARY KEY,
    cliente VARCHAR(10) NOT NULL,
    data DATE,
    FOREIGN KEY(cliente) REFERENCES cliente(nome) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE dettaglio_ordine (
    numero_ordine INT,
    codice_articolo VARCHAR(10),
    quantita INT NOT NULL DEFAULT 1,
    PRIMARY KEY(numero_ordine, codice_articolo),
    FOREIGN KEY(numero_ordine) REFERENCES ordine(numero) ON DELETE CASCADE ON UPDATE
CASCADE,
    FOREIGN KEY(codice_articolo) REFERENCES articolo(codice) ON DELETE CASCADE ON
UPDATE CASCADE
);
```

scrivere i comandi SQL che permettono di:

B.3.a. Ottenere la lista ordinata dei clienti in ordine crescente per nome.

B.3.b. Ottenere il costo totale per gli ordini numero 1 e 3, distinta ciascun ordine.

B.3.c. Ottenere la lista, con numero e cliente, degli ordini, che siano stati effettuati dopo il 31 dicembre 2012 oppure per i quali non sia specificata la data.

B.3.d. Aggiornare nell'intero database il cliente ('ACME srl', 'Via Garibaldi 22', 'Verona'), che si è trasferito in Via Meucci 12 a Roma e il cui nome è ora 'ACME Spa'.

B.3.e. Rimuovere gli ordini effettuati da clienti che hanno effettuato meno di 2 ordini.

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 (se n è 0 l'attributo è opzionale) che può avere una persona e m il numero massimo. Normalmente la cardinalità (1,1) si omette.

A.2. Possiamo pensare ad un dato come ad un dato "grezzo", cioè non elaborato, non interpretato, mentre l'informazione è il dato interpretato (o elaborato), che arricchisce la conoscenza dell'interprete. Ad esempio il dato numerico 37845983 diventa un'informazione nel momento in cui lo interpreto ad es. come numero di telefono di un certa persona oppure diventa un'altra informazione se lo interpreto come il numero di abitanti della Spagna al 31/12/1985.

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. L'attributo titolo è opzionale (0) e al massimo vi può essere un solo titolo per IMPIEGATO.

B.1.b. Tra IMPIEGATO e UFFICIO esiste una doppia relazione binaria. Un'occorrenza di IMPIEGATO partecipa sempre alla relazione ASSEGNAZIONE con con una sola occorrenza dell'entità UFFICIO, cioè un impiegato è assegnato ad uno ed un solo ufficio. Inoltre, un ufficio potrebbe essere sprovvisto di impiegati. Ogni ufficio ha al massimo un capoufficio, ma potrebbe anche non averne. Un impiegato può dirigere eventualmente un solo un ufficio.

B.1.c. L'attributo nome di UFFICIO da solo non è in grado di identificare ogni occorrenza di ufficio: potrebbero esistere uffici con lo stesso nome in sedi diverse. Quindi, per poter identificare ciascun ufficio si ricorre ad un identificatore esterno,utilizzando il legame di LOCALIZZAZIONE con l'entità SEDE. Per avere un identificare valido abbiamo bisogno della cardinalità (1,1), in quanto ogni ufficio deve essere localizzato in una ed una sola sede per far sì, appunto, che nome e sede siano univoci.

B.1.d. Vi sono due modalità semplici: aggiungo un attributo a IMPIEGATO oppure ad ASSEGNAZIONE. Questo secondo caso risulta essere più "elegante" in quanto traduce meglio il concetto che un impiegato viene assegnato ad una data sede a partire da una certa data.

B.1.e. `IMPIEGATO(matricola, cognome, nome, titolo*, ufficio)`
`UFFICIO(nome, sede, direttore*)`
`SEDE(nome)`
con vincolo di integrità referenziale tra l'attributo `ufficio` dell'entità `IMPIEGATO` e l'entità UFFICIO, tra l'attributo `direttore` di `UFFICIO` e l'entità `IMPIEGATO` e tra l'attributo `sede` di `UFFICIO` e l'entità `SEDE`.

B.2.

B.2.a.

```
CREATE TABLE citta (
    nome VARCHAR(32) PRIMARY KEY
);
CREATE TABLE squadra (
    nome VARCHAR(32) PRIMARY KEY,
    anno_fondazione INT,
    citta VARCHAR(32) NOT NULL,
    FOREIGN KEY(citta) REFERENCES citta(nome) ON UPDATE CASCADE ON DELETE NO ACTION
);
CREATE TABLE atleta (
    nome VARCHAR(32),
    cognome VARCHAR(32),
    data_nascita DATE,
    squadra VARCHAR(32) NOT NULL,
    PRIMARY KEY(nome, cognome),
    FOREIGN KEY(squadra) REFERENCES squadra(nome) ON UPDATE CASCADE ON DELETE NO ACTION
);
CREATE TABLE partita (
    squadra1 VARCHAR(32),
    squadra2 VARCHAR(32),
    data DATE,
    PRIMARY KEY(squadra1, squadra2),
    FOREIGN KEY(squadra1) REFERENCES squadra(nome) ON UPDATE CASCADE ON DELETE NO ACTION,
    FOREIGN KEY(squadra2) REFERENCES squadra(nome) ON UPDATE CASCADE ON DELETE NO ACTION
);
```

B.2.b.

cognome
Cabrini
Collovati
Conti

B.2.c.

nome	citta
Juventus	Torino
Inter	Milano

B.2.d. In base ai vincoli precedenti, per eliminare una squadra devo prima eliminare i suo eventuali giocatori e le partite a cui ha partecipato.

```
DELETE FROM atleta WHERE squadra = 'Juventus';
DELETE FROM partita WHERE squadra1 = 'Juventus' OR  squadra2 = 'Juventus';
DELETE FROM squadra WHERE nome = 'Juventus';
```

B.2.e. I comandi sono corretti. Dopo la loro esecuzione PARTITA e SQUADRA avranno entrambe 3 record.

B.3.

B.3.a. `SELECT * FROM cliente ORDER BY nome;`

B.3.b. `SELECT SUM(quantita * costo_unitario) AS totale_ordine FROM dettaglio_ordine INNER JOIN articolo ON codice_articolo = codice WHERE numero_ordine IN (1, 3) GROUP BY numero_ordine;`

B.3.c. `SELECT numero, cliente FROM ordine WHERE data > '2012-12-31' OR data IS NULL;`

B.3.d. `UPDATE cliente SET nome = 'ACME Spa', indirizzo = 'Via Meucci 12', citta = 'Roma' WHERE nome = 'ACME srl';`

B.3.e. `DELETE FROM ordine WHERE cliente IN (SELECT nome FROM cliente INNER JOIN ordine ON nome = cliente GROUP BY nome HAVING COUNT(*) < 2);`