

Soluzioni di alcuni degli esami passati di Basi di Dati. Alla fine del file inserir anche soluzioni di alcuni esercizi di ripasso (Ripasso.pdf) che sono stati condivisi dal Prof.

Mattia Frega .

Esame Febbraio 2020

Esercizio 1

A)

La query produce in output due colonne , una con i nomi degli stati e una con un campo che conta quante catene montuose si estendono in quello stato.

Considerando che le catene montuose sono raggruppate per Stato e che al più la tabella estensione ha 50 righe, supponendo Stati tutti diversi avremo al massimo 50 righe

Al minimo avremo 2 righe: considerando 40 catene montuose ma una tabella di estensione di 50 righe vuol dire che, al minimo, 40 catene montuose si estenderanno su uno stato, ma 10 di queste si estendereanno anche su un altro stato (non sono possibili duplicati) e quindi al minimo avremo 2 stati, di conseguenza 2 righe sulla query

B)

```
SELECT DISTINCT NomeStato
FROM CATENAM as C1 JOIN ESTENSIONE AS E1 JOIN ESTENSIONE AS E2
ON (C1.Nome = E1.NomeCatena AND C1.Nome = E2.NomeCatena)
WHERE (E1.NomeStato <> E2.NomeStato)
```

C)

```
SELECT COUNT(*) as Totale
FROM CATENAM as C1
WHERE (C1.Altitudine>2000) AND T1.Nome IN(
    SELECT NomeCatena
    FROM ESTENSIONE as E1
    WHERE (ESTENSIONE.NomeStato = "Italia") OR
    (ESTENSIONE.NomeStato = "Germania") OR
    (ESTENSIONE.NomeStato = "Svizzera")) SBAGLIATO
```

CORRETTO :

Select count(*) as NumeroCatene

From Catenam

Where altitudine > 2000 and nome in (select NomeCatena from Estensione where
nomeStato="Italia" and NomeCatena in
(select NomeCatena from Estensione where
nomeStato="Svizzera" and NomeCatena in
(select NomeCatena from Estensione where
nomeStato="germania"))

D)

SELECT STATO.Nome, STATO.Superficie

FROM ESTENSIONE, STATO

WHERE (ESTENSIONE.NomeStato = CATENAM.Nome) AND (CATENA.Nome = "Alpi
Carniche") AND (NOT(ESTENSIONE.NomeCatena = "Alpi Giulie")) SINTASSI ERRATA

Select Nome, superficie

From ESTENSIONE, STATO

Where NomeStato=Nome and NomeCatena="Alpi carniche" and NomeStato NOT
IN(select NomeStato from ESTENSIONE where NomeCatena="Alpi Giulie")

E)

SELECT Continente

FROM CATENAM AS C, ESTENSIONE AS E, STATO AS S

WHERE (C.Lunghezza = 1000) AND (C.Nome = E.NomeCatena) AND (S.Nome =
E.NomeStato)

GROUP BY Continente

HAVING (COUNT(*)>5)

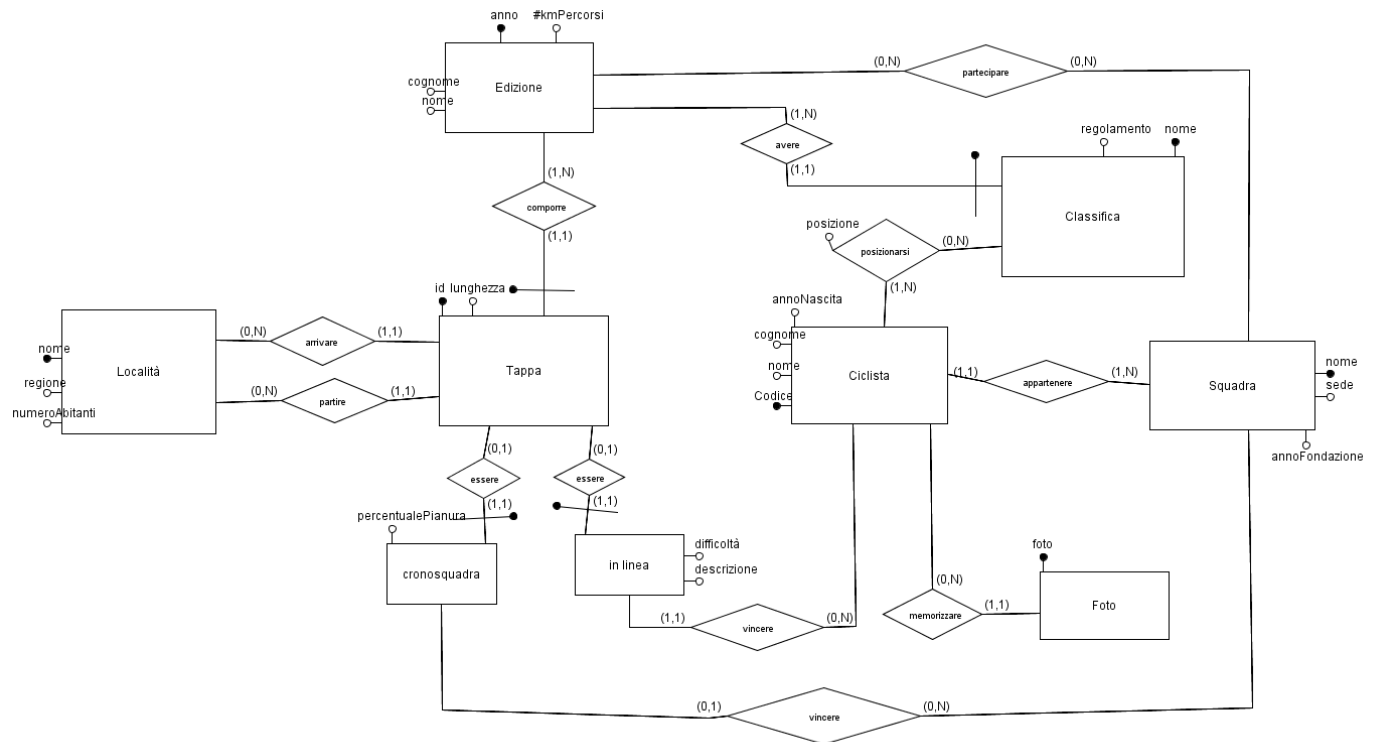
F)

db.CATENAM.find({altitudine:{\$gte:2000}, lunghezza:{\$lte:300}}, {Nome:1, _id:0})

L'ultima parte della query ({Nome:1}) governa quali attributi saranno mostrati nella query risultante. In questo caso, solo il nome. Usare \$gt e \$lt al posto di \$gte e \$lte in caso di strettamente minore.

Esercizio 2

A)



B)

La traduzione che minimizzi i valori NULL per le generalizzazioni la soluzione 3, cio sostituire la generalizzazione con entit genitore e figli, inserendo al posto della generalizzazione una relazione di questo tipo

GENITORE (0,1) - (1,1) FIGLIO

Aggiungendo in pi l'identificatore esterno verso genitore alle entit figli

Schema Logico:

EDIZIONE(Anno, Cognome, Nome, #kmPercorsi)

SQUADREPARTICIPANTI(AnnoEdizione, NomeSquadra)

SQUADRA(Nome, Sede, annoFondazione)

TAPPA(id, Lunghezza, AnnoEdizione, NomeLocalit Arrivo, NomeLocalit Partenza)

LOCALITA(Nome, Regione, NumeroAbitanti)

CRONOSQUADRA(percentualePianura, idTappa, NomeSquadraVincitrice)

IN LINEA(difficolt , descrizione, idTappa, CodiceCiclistaVincitore)

CICLISTA(Codice, Nome, Cognome, AnnoNascita, NomeSquadra)

CLASSIFICA(Nome, Regolamento, AnnoEdizione)

POSIZIONICLASSIFICA(NomeClassifica, CodiceCiclista, AnnoEdizione, Posizione)

FOTO(Foto, CodiceCiclista)

Vincoli di integrità referenziale:

SQUADREPARTECIPANTI.AnnoEdizione -> EDIZIONE.Anno

SQUADREPARTECIPANTI.NomeSquadra -> SQUADRA.Nome

TAPPA.AnnoEdizione -> EDIZIONE.Anno

TAPPA.NomeLocalit Arrivo -> LOCALITA.Nome

TAPPA.NomeLocalit APartenza -> LOCALITA.Nome

CRONOSQUADRA.idTappa -> TAPPA.id

IN LINEA.idTappa -> TAPPA.id

CICLISTA.NomeSquadra -> SQUADRA.Nome

CLASSIFICA.AnnoEdizione -> EDIZIONE.Anno

FOTO.CodiceCiclista -> CICLISTA.Codice

POSIZIONICLASSIFICA.NomeClassifica -> CLASSIFICA.Nome

POSIZIONICLASSIFICA.CodiceCiclista -> CICLISTA.Codice

POSIZIONICLASSIFICA.AnnoEdizione -> EDIZIONE.Anno

C)

Formula per il calcolo del costo =

=frequenza * coefficiente * (letture + peso * scritture)

$C(\text{op1 rid}) = 1 * 0.5 * (0 + 2 * 3)$

$C(\text{op2 rid}) = 5 * 1 * (20 + 20 + 0)$

$C(\text{op1}) = 1 * 0.5 * (0 + 2 * 2)$

$C(\text{op2}) = 5 * 1 * (20 + 20 + 0)$

Speedup = $C(s)/(C \text{ srid})$

Occupazione di memoria

Assumendo che il campo #kmPercorsi sia implementato con un intero che occupa 4 bytes ed essendoci una edizione all'anno, l'occupazione di memoria è trascurabile.

Esercizio 3

A)

Individuo come superchiave ad occhio (AnnoEdizione, NomeCantante). Calcolo la chiusura **{AnnoEdizione, NomeCantante}+F**

={AnnoEdizione, NomeCantante, Nome, Presentatore, Città, NumeroPartecipazioni, NomeTeatro, NumeroVittorie, TitoloCanzone, Durata} -> sono tutti gli attributi della tabella quindi una superchiave.

Verifico se è anche chiave controllando la chiusura di AnnoEdizione e NomeCantante singolarmente

{AnnoEdizione}+F

={AnnoEdizione, Nome, Presentatore, Città, NomeTeatro} -> non sono tutti gli attributi della tabella quindi non può essere chiave

{NomeCantante}+F

={NomeCantante, NumeroPartecipazioni} -> non sono tutti gli attributi della tabella quindi non può essere chiave

Di conseguenza la chiave {AnnoEdizione, NomeCantante}

B)

Dipendenze funzionali:

1 AnnoEdizione NomePresentatore Città

2 NomeCantante NumeroPartecipazioni

3 TitoloCanzone Durata

4 Città NomeTeatro

5 NomeCantante NumeroVittorie

6 AnnoEdizione NomeCantante TitoloCanzone

FNBC: per ogni $X \rightarrow Y$ x superchiave

1,2,3,4,5 non rispettano la condizione

3FN: per ogni $X \rightarrow Y$ x superchiave OPPURE Y parte della chiave

3,4 non rispettano la condizione (anche 1,2,5)

2FN: se non ci sono dipendenze parziali

1,2,5 sono dipendenze parziali

Esercizio 4

A)

L'algoritmo di Naïve Bayes stima la probabilità di una istanza di appartenere ad una certa classe. Fa uso della tecnica statistica della probabilità condizionata

B)

UNDO = {T2, T6}

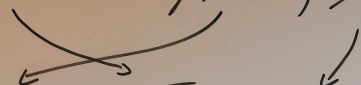
REDO = {T4, T1, T5}

Procedimento:

ultimo CK \rightarrow CK(T_1, T_2)

UNDO = $\{\cancel{I_1}, T_2, \cancel{I_4}, \cancel{I_5}, T_6\}$

REDO = $\{T_4, T_1, T_5\}$



Esame 22 gennaio 2016

Esercizio 1

A)

```
SELECT Titolo, Genere
FROM FILM
WHERE (Titolo NOT IN
      (SELECT TitoloF
       FROM NOLEGGIO
       WHERE (Data BETWEEN 1/1/2016 AND 21/1/2016)))
```

B)

```
SELECT AVG(Eta)
FROM UTENTE
WHERE(Citta = "Bologna" AND
      NumTesserata IN(
        SELECT DISTINCT NumT
        FROM NOLEGGIO
        WHERE Data BETWEEN 1/1/2016 AND 21/1/2016
      )
)
```

C) //incerta

```
CREATE VIEW totaleFilmNoleggiatiPerGenere(Genere,NumT,TOT) AS(
SELECT Genere, NumT, COUNT(*) AS TOT
FROM FILM JOIN NOLEGGIO ON Titolo = TitoloF
GROUP BY (Genere, NumT)
)
```

```

SELECT Genere, Nome, Cognome
FROM totaleFilmNoleggiatiperGenere JOIN UTENTE
ON NumT = NumTessera
WHERE TOT = (SELECT MAX(TOT) FROM totaleFilmNoleggiatiperGenere)
GROUP BY Genere

```

D)

```

CREATE TABLE NOLEGGIO(
    NumT integer,
        TitoloF varchar(50),
        dataN date NOT NULL,
        note varchar(2000) DEFAULT "nessun commento",

    PRIMARY KEY(NumT, TitoloF),
    FOREIGN KEY (NumT) REFERENCES UTENTI(NumTessera),
    FOREIGN KEY (TitoloF) REFERENCES FILM(Titolo)

```

```

        CHECK(Count(NumT, date))<=3

```

```

)ENGINE = INNODB;

```

```

        CREATE ASSERTION (CHECK(Count(NumT, date))<=3
    )

```

E)

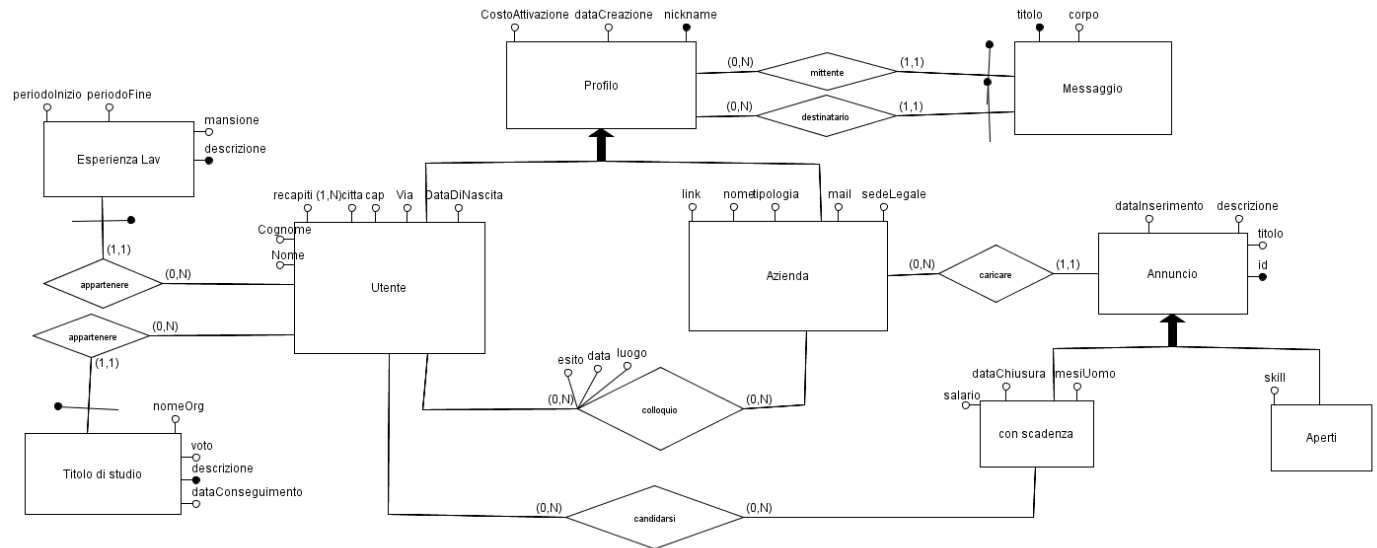
```

Db.FILM.find({StatoCopia = "Eccellente", Anno = "2014"},{Titolo:1, Genere:1})

```

Esercizio 2

A)



Sicuramente esistono diverse soluzioni giuste:

- Nello schema ER sopra ho scelto la primary key che mi sembrava più adatta, ma si poteva sceglierne altre o mettere campi id.
- Le generalizzazioni sono senza primary key: prenderanno identificatore esterno durante la traduzione in schema logico.
- Colloquio modellato con una relazione preclude il fatto che un utente possa fare più volte un colloquio con la stessa azienda. Modellando anche questa eventualità sarebbe servita una entità Colloquio in cui mettere primary key anche data.
- Per esperienza di Lavoro e Titolo di studio ho aggiunto un identificatore esterno, ma non era esplicitamente richiesto dal testo. Ho scelto descrizione come primary key, ma probabilmente sarebbe stato meglio scegliere i periodi.
- Via/città/cap si può modellare come attributo composto ma non ho trovato l'impostazione su JDER. Comunque equivalente.

B)

La soluzione che minimizzi i valori null nelle tabelle è la soluzione 3, ovvero sostituire le entità figlie della generalizzazione con delle entità in relazione 0,1 - 1,1 con il genitore e aggiungere l'identificatore esterno.

Schema Logico:

PROFILO(nickname, dataCreazione, costoAttivazione)

MESSAGGIO(titolo, corpo, nicknameMittente, nicknameDestinatario)

UTENTE(Nome, Cognome, città, cap, via, DataDiNascita, nicknameProfilo)

RECAPITO(numerotelefono, nicknameUtente)

AZIENDA(link, nome, tipologia, mail, sedeLegale, nicknameAzienda)

ANNUNCIO(id, titolo, descrizione, dataInserimento, nicknameAzienda)

CONSCANDENZA(salario, dataChiusura, mesiUomo, idAnnuncio)

APERTI(skill, idAnnuncio)

ESPERIENZALAV(descrizione, mansione, periodiFine, periodilnizio, nicknameUtente)

TITOLOSTUDIO(descrizione, nomeOrg, voto, dataConseguimento, nicknameUtente)

CANDIDATURE(idAnnuncio, nicknameUtente)

COLLOQUIO(nicknameUtente, nicknameAzienda, esito, data, luogo)

Vincoli di integrità referenziale:

MESSAGGIO.nicknameDestinatario -> PROFILO.nickname

MESSAGGIO.nicknameMittente -> PROFILO.nickname

UTENTE.nicknameProfilo -> PROFILO.nickname

AZIENDA.nicknameProfilo -> PROFILO.nickname

RECAPITO.nicknameUtente -> UTENTE.nicknameProfilo

ESPERIENZALAV.nicknameUtente -> UTENTE.nicknameProfilo

TITOLODISTUDIO.nicknameUtente -> UTENTE.nicknameProfilo

CANDIDATURE.nicknameUtente -> UTENTE.nicknameProfilo

ANNUNCIO.nicknameAzienda -> AZIENDA.nicknameProfilo

COLLOQUIO.nicknameUtente -> UTENTE.nicknameProfilo

COLLOQUIO.nicknameAzienda -> AZIENDA.nicknameProfilo

CONSCADENZA.idAnnuncio -> ANNUNCIO.id

APERTI.idAnnuncio -> ANNUNCIO.id

CANDIDATURE.idAnnuncio -> ANNUNCIO.id

C)

$$C(\text{op1}) = 50 * 1 * (2*3) = 300$$

$$C(\text{op2}) = 2 * 1 * (1+4+4+2+2) = 26$$

$$C(\text{op3}) = 10 * 0.5 * (2*(20+20)) = 400$$

Esercizio 3

A)

$$\{ABD\}+F = \{A,B,D,C,E\}$$

quindi ABD superchiave, scopro se minimale:

$$\{AB\}+F = \{A,B,C,D,E\}$$

$$\{BD\}+F = \{B,D,A,E,C\}$$

$$\{DA\}+F = \{D,A\}$$

Quindi sia AB che BD sono superchiavi, scopro se sono minimali:

$$\{A\}+F = \{A\}$$

$$\{B\}+F = \{B,A,D,C,E\}$$

Quindi la chiave B

B)

Dipendenze funzionali:

$$1 \ AB \rightarrow C$$

$$2 \ BD \rightarrow AE$$

$$3 \ B \rightarrow AD$$

FNBC [per ogni $X \rightarrow Y$ x superchiave]

S perch la 1 ha a sinistra una superchiave (dimostrato sopra), la 2 anche e la 3 ha a sinistra una chiave che per definizione superchiave e quindi va bene

3FN [per ogni $X \rightarrow Y$ x superchiave o Y parte della chiave]

S perch in FNBC che una forma pi stringente di 3FN quindi sicuramente rispetta le condizioni della 3FN

Esercizio 4

A)

L'algoritmo delle K-medie un algoritmo di clusterizzazione non gerarchico che richiede di indicare in anticipo il numero di cluster che si vuole creare. Tramite l'uso di punti medi calcola l'appartenenza o meno di una certa istanza al cluster e sceglie quello pi vicino.

B)

Ultimo CK = CK(T0,T1,T3)

UNDO = {~~T0~~, T1, T3, T4, ~~T5~~}

REDO = {T0, T5}

Esame 1

Esercizio 1

A)

Colonne: 1 (quella del count)

Righe: 0 al minimo se nessuna nazionale stata fondata dopo il 1960

20 al massimo

B)

```
SELECT DISTINCT (Anno)
```

```
FROM EDIZIONE
```

```
WHERE Numero IN (
```

```
    SELECT NumeroEdizione
```

```
    FROM PARTECIPAZIONE
```

```
    WHERE NomeNazione IN (
```

```
        SELECT Nome
```

```
        FROM NAZIONALE
```

```
        WHERE Continente = Oceania))
```

C)

```
SELECT Nome, AnnoFondazione
```

```
FROM NAZIONALE
```

```
WHERE Nome NOT IN (
```

```
    SELECT NomeNazionale
```

```
    FROM PARTECIPAZIONE JOIN EDIZIONE ON (NumeroEdizione = Numero)
```

```
    WHERE NazionaleVincitrice = NomeNazionale
```

```
    AND Continente = Europa AND Anno > 1950)
```

D)

```
CREATE VIEW numeroNazionaliEuropeePerEdizione (NumEdz, contatore) as(
```

```
    SELECT NumeroEdizione, COUNT(*) as contatore
```

```
FROM PARTECIPAZIONE JOIN NAZIONALE ON Nome = NomeNazionale
```

WHERE Continente = EUROPA

GROUP BY NumeroEdizione

)

SELECT Numero, Anno

FROM EDIZIONE JOIN numeroNazionaliEuropeePerEdizione ON NumEdz = Numero

WHERE Numero =(SELECT (MIN(NumEdz))

FROM numeroNazionaliEuropeePerEdizione)

E) //incerta

SELECT Continente

FROM NAZIONALE JOIN PARTECIPAZIONE JOIN EDIZIONE

ON (Numero = NumeroEdizione) AND (Nome = NomeNazionale)

WHERE Anno >=1980

GROUP BY Continente

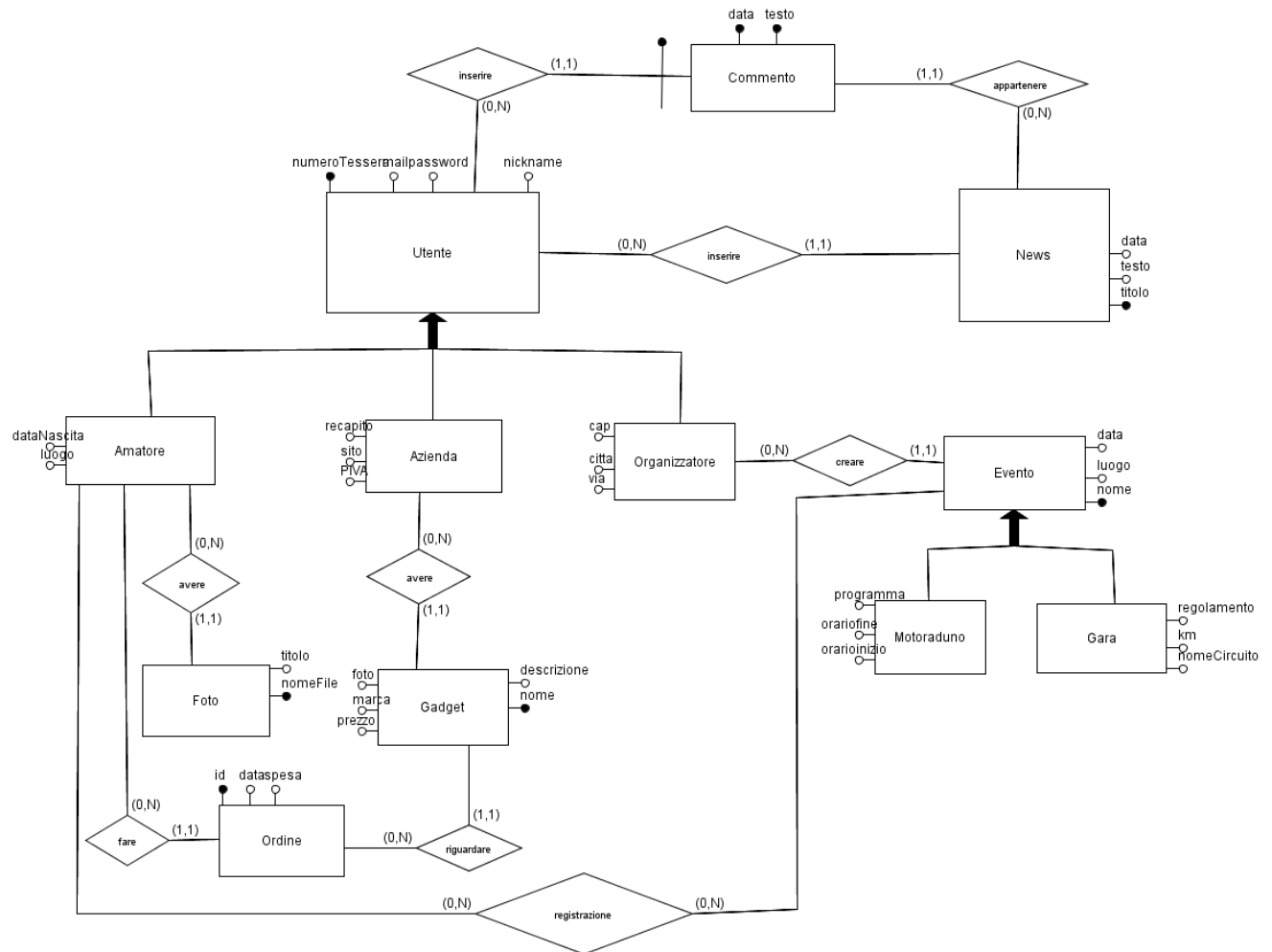
HAVING COUNT(*)>=2

F)

Db.Nazionale.find({\$or{Continente:Europa},{AnnoFondazione<1930}},{Nome:1, _id:0})

Esercizio 2

A)



Nota che:

- La chiave esterna su Commento non è specificata dal testo, ma modella anche quel caso in cui due utenti inserissero lo stesso identico messaggio alla stessa identica data, quindi ho pensato di metterlo

B)

La traduzione che minimizzi la presenza di valori NULL è la seguente:

UTENTE(numeroTesser, mail, password, nickname)

NEWS(titolo, testo, data, tesseraUtente)

COMMENTO(data, testo, tesseraUtente, titoloNews)

AMATORE(tesseraUtente, dataNascita, luogo)

AZIENDA(tesseraUtente, PIVA, sito, recapito)

ORGANIZZATORE(tesseraUtente, cap, via, città)

FOTO(nomeFile, titolo, tesseraUtenteAmatore)
 GADGET(nome, descrizione, foto, marca, prezzo, tesseraUtenteAzienda)
 ORDINE(id, data, spesa, tesseraUtenteAmatore, tesseraUtenteAzienda)
 EVENTO(nome, luogo, data, tesseraUtenteOrganizzatore)
 MOTORADUNO(programma, orarioFine, orarioInizio, nomeEvento)
 GARA(regolamento, km, nomeCircuito, nomeEvento)
 REGISTRAZIONI(nomeEvento, tesseraUtenteAmatore)

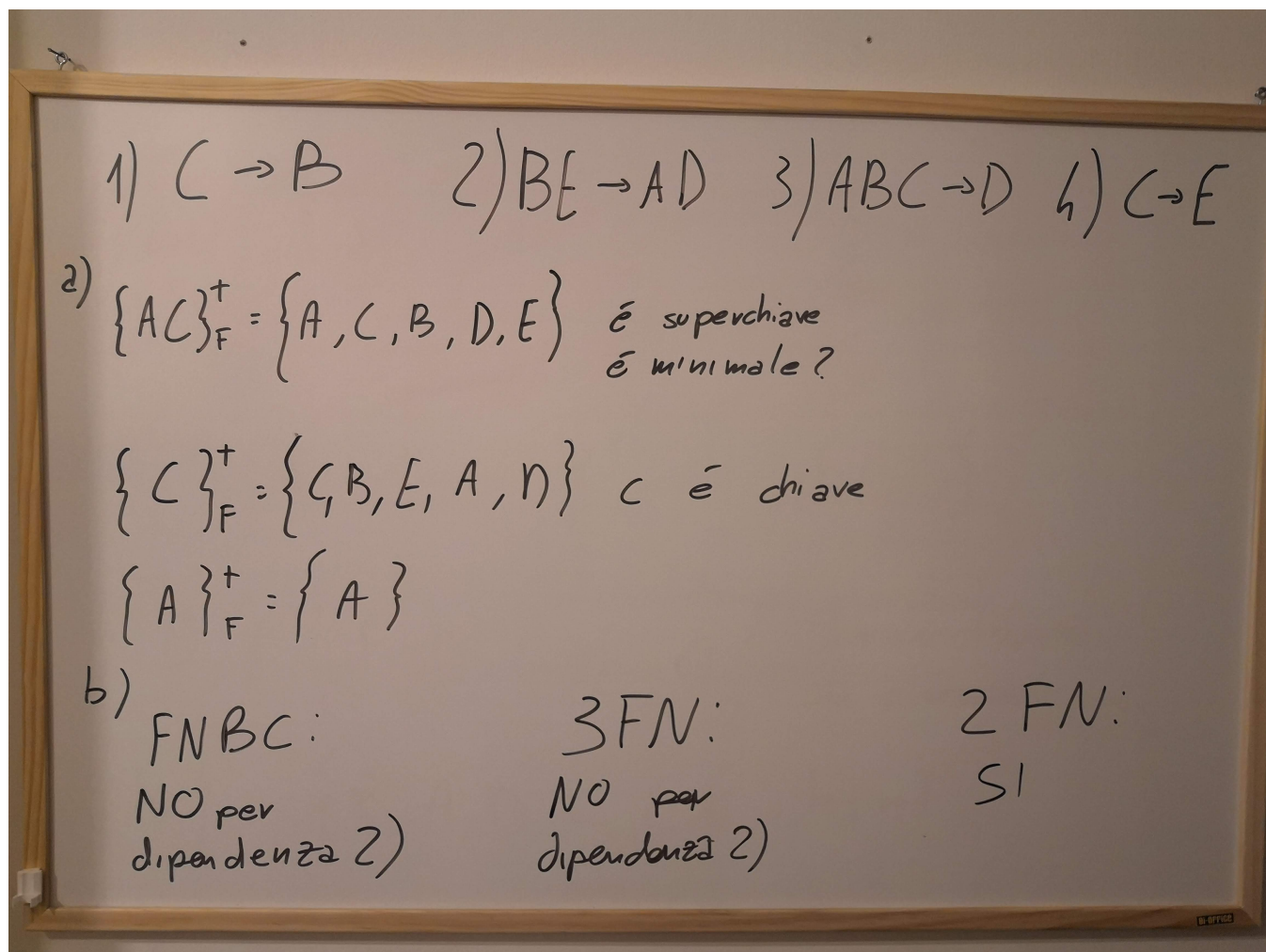
Vincoli di integrità referenziale:

COMMENTO.tesseraUtente -> UTENTE.Tessera
 COMMENTO.titoloNews -> NEWS.Titolo
 AMATORE.tesseraUtente -> UTENTE.tessera
 AZIENDA.tesseraUtente -> UTENTE.tessera
 ORGANIZZATORE.tesseraUtente -> UTENTE.tessera
 FOTO.tesseraUtenteAmatore -> AMATORE.tesseraUtente
 GADGET.tesseraUtenteAzienda -> AZIENDA.tesseraUtente
 ORDINE.tesseraUtenteAmatore -> AMATORE.tesseraUtente
 ORDINE.tesseraUtenteAzienda -> AZIENDA.tesseraUtente
 EVENTO.tesseraUtenteOrganizzatore -> ORGANIZZATORE.tesseraUtente
 MOTORADUNO.nomeEvento -> EVENTO.nome
 GARA.nomeEvento -> EVENTO.nome
 REGISTRAZIONI.nomeEvento -> EVENTO.nome
 REGISTRAZIONI.tesseraUtenteAmatore -> AMATORE.tesseraUtente

C)

- $10 * 1 * (2^{*(1+1)})$
- $2 * 1 * (2^{*0+1+10+10})$
- $3 * 0.5 * (2^{*(3)})$

Esercizio 3



Esercizio 4

A)

Il teorema CAP uno dei teoremi principali riguardante i sistemi distribuiti. Esso afferma che un sistema distribuito pu avere 3 caratteristiche fondamentali:

- Consistency: tutti i nodi della rete vedono gli stessi dati
- Availability: capacit di rendere il servizio sempre disponibile
- Partition Tolerance: il servizio continua a funzionare in maniera corretta anche se accade che un nodo della rete vada offline.

In fase di progettazione del sistema si potranno scegliere solo 2 di queste caratteristiche, e mai 3 contemporaneamente. Esempio: un sistema che offre partition tolerance potr essere o consistente o

disponibile, non entrambi.

B)

Two Phase Lock: una transazione, prima di iniziare a fare operazioni sui dati, acquisisce tutti i lock degli oggetti di cui ha bisogno per lavorare.

Strict Two Phase Lock: come la 2PL ma in più i lock sono rilasciati solo DOPO aver fatto commit (o abort).

T0	T1
Lock0(x)	
W0(x)	
	Lock1(y)
	W1(y)
	Unlock1(y)
Unlock0(x)	
	Lock1(x)
	R1(x)
	Unlock1(x)
Commit0	
	Commit1

Lo schedule non rispetta il 2PL: T0 necessita di lavorare con l'oggetto x e ne acquisisce il lock subito, ma T1 acquisisce solo il lock di y prima di iniziare a lavorare, mentre dovrebbe aspettare anche quello di x.

Lo schedule non rispetta la S2PL in quanto i lock sono rilasciati prima del commit, e comunque perché è un protocollo più stringente di 2PL.

Esame Luglio 2016

Esercizio 1

A)

```
SELECT Codice, Nome, Cognome
FROM GIOCATORE
WHERE (AnnoNascita < 1988) AND
      Codice NOT IN (SELECT CodiceGiocatore
                     FROM CONTRATTO
                     WHERE Anno = 2013)
```

B)

```
SELECT COUNT(*)
FROM SQUADRA
WHERE SQUADRA.Citta = BOLOGNA
      AND SQUADRA.Nome IN
      (SELECT NomeSquadra
       FROM CONTRATTO
       WHERE CodiceGiocatore IN
              (SELECT Codice
               FROM GIOCATORE
               WHERE Nome = Andrea AND Cognome = Rossi))
```

C)

```
CREATE VIEW sommaIngaggiPerSquadra (NomeSquadra, SommaIngaggio) AS(
SELECT NomeSquadra, SUM(Ingaggio)
FROM SQUADRA, CONTRATTO, GIOCATORE
WHERE (ANNO = 2015) AND (SQUADRA.Nome = CONTRATTO.NomeSquadra)
AND (GIOCATORE.Codice = CONTRATTO.CodiceGiocatore) AND
(GIOCATORE.AnnoNascita >= 1990)
```

GROUP BY NomeSquadra

)

SELECT Nome, Citta

FROM SQUADRA, sommaIngaggiPerSquadra

WHERE SQUADRA.Nome = sommaIngaggiPerSquadra.NomeSquadra

AND SommaIngaggio = (SELECT MAX(SommaIngaggio)

FROM sommaIngaggiPerSquadra)

D)

CREATE TABLE CONTRATTO (

Id varchar(10),

CodiceGiocatore varchar(15),

NomeSquadra varchar(50),

Anno varchar(4) NOT NULL,

Ingaggio integer,

CHECK (Id LIKE ID_%000),

CHECK (COUNT(CodiceGiocatore,Anno)<=3),

PRIMARY KEY (Id),

FOREIGN KEY CodiceGiocatore REFERENCES GIOCATORE(Codice),

FOREIGN KEY NomeSquadra REFERENCES SQUADRA(Nome)

)

E)

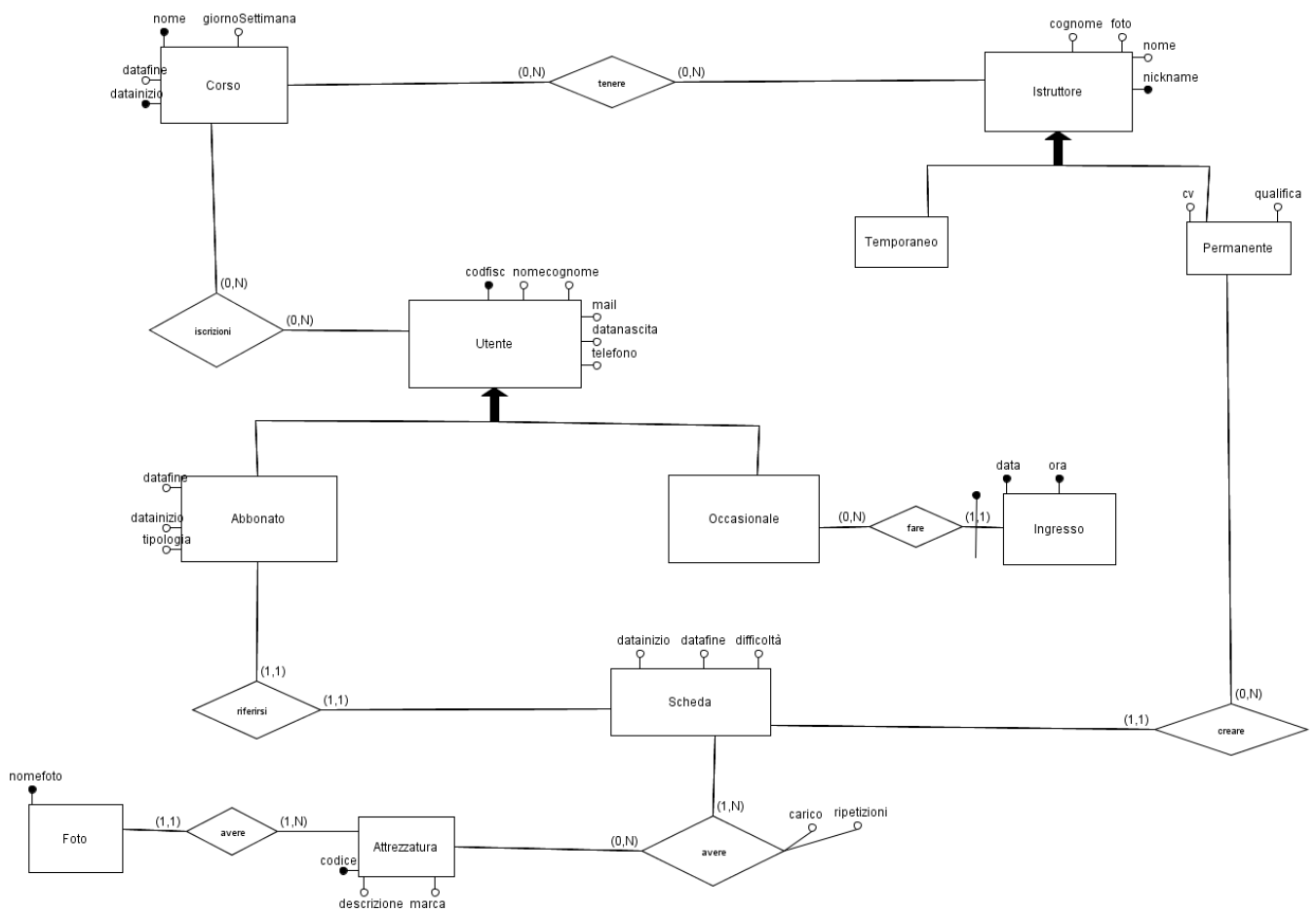
Db.GIOCATORE.find(

{Nome:Mario, Cognome:Rossi, AnnoNascita:1990},{codice:1}

)

Esercizio 2

A)



Note:

- Foto poteva essere modellato come un attributo multiplo, tanto poi diventa tabella
- Nella relazione riferirsi tra Abbonato e Scheda mi sembra più logico mettere Abbonato(0,1)-(1,1)Scheda, ma il testo dice che un Abbonato dispone di una scheda e non "può disporne di"
- In ingresso obbligatorio l'identificatore esterno: due utenti occasionali potrebbero entrare alla stessa data e alla stessa ora.

B)

Come al solito, la soluzione che minimizza i valori NULL tradurre le generalizzazioni con la soluzione 3.

Schema Logico:

CORSO(nome, dataInizio, dataFine, giornoSettimana)

ISTRUTTORE(nickname, nome, foto, cognome)

TENERECORSI(nicknameIstruttore, nomeCorso, dataInizio)
PERMANENTE(nicknameIstruttore, cv, qualifica)
TEMPORANEO(nicknameIstruttore)
UTENTE(codfisc, nome, cognome, mail, dataNascita, telefono)
ISCRIZIONI(codfiscUtente, nomeCorso, dataInizioCorso)
ABBONATO(codfiscUtente, dataFine, dataInizio, tipologia)
OCCASIONALE(codfiscUtente)
INGRESSO(data, ora, codfiscUtenteOccasionale)
SCHEDA(dataInizio, dataFine, difficoltà, codfiscUtenteAbbonato, nicknameIstruttorePermanente)
ATTREZZATURA(codice, descrizione, marca)
ATTREZZATURESCHEDE(codiceAttrezzatura, dataInizioScheda, codfiscScheda, carico, ripetizioni)
FOTO(nomeFoto, codiceAttrezzatura)

Vincoli di integrità

TENERECORSI.nicknameIstruttore -> ISTRUTTORE.nickname
TENERECORSI.nomeCorso -> CORSO.nome
TENERECORSI.dataInizio -> CORSO.dataInizio
PERMANENTE.nicknameIstruttore -> ISTRUTTORE.nickname
TEMPORANEO.nicknameIstruttore -> ISTRUTTORE.nickname
ISCRIZIONI.codfiscUtente -> UTENTE.codFisc
ISCRIZIONI.nomeCorso -> CORSO.nome
ISCRIZIONI.dataInizioCorso -> CORSO.dataInizio
INGRESSO.codfiscUtenteOccasionale -> OCCASIONALE.codfiscOccasionale
SCHEDA.codfiscUtenteAbbonato -> ABBONATO.
SCHEDA.nicknameIstruttorePermanente -> PERMANENTE.nicknameIstruttore
ATTREZZATURASCHEDE.codiceAttrezzatura -> ATTREZZATURA.codice
ATTREZZATURASCHEDE.dataInizioScheda -> SCHEDA.dataInizio
ATTREZZATURASCHEDE.codfiscScheda -> SCHEDA.nicknameIstruttorePermanente
FOTO.codiceAttrezzatura -> ATTREZZATURA.codice

C)

- $20 * 1 * (2 * (1+1))$
- $1 * 1 * (10+10)$
- $2 * 1 * (30)$

Esercizio 3

A)

Dipendenza funzionali:

1. $C \rightarrow AB$
2. $CA \rightarrow D$
3. $BF \rightarrow E$
4. $F \rightarrow D$

$\{CF\} + F = \{C, F, A, B, D, E\}$ quindi $\{C, F\}$ superchiave, controllo se minimale

$\{C\} + F = \{C, A, B, D\}$ non chiave

$\{F\} + F = \{F, D\}$ non chiave

Quindi $\{CF\} + F$ minimale e di conseguenza chiave

B)

FNBC = No, le condizioni 1-2-3-4 non rispettando la condizione $X \rightarrow Y$ X superchiave

3FN = No, le condizioni 1-2-3-4 non rispettando la condizione $X \rightarrow Y$ X superchiave OPPURE Y una parte di chiave

2FN = No, ci sono dipendenze parziali

Esercizio 4

A)

Uno Schedule seriale uno schedule in cui le operazioni di ciascuna transazione appaiono in sequenza una dopo l'altra, ovvero non sono inframezzate da operazioni di altre transazioni:

$\{\text{operazioni } T0\}$ POI $\{\text{operazioni } T1\}$ POI $\{\text{operazioni } T2\}$

Uno schedule serializzabile uno schedule che produce lo stesso risultato di uno schedule seriale, ma nel quale le operazioni di una transazioni possono essere inframezzate da altre transazioni e controllate tramite un lock manager

B)

Ultimo CK = CK(T0, T1, T3)

UNDO = {T1, T3, T5}

REDO = {T0, T4}

Esame Settembre 2016

Esercizio 1

A)

```
SELECT email, COUNT(*)  
FROM STUDENTE, MAILINGLIST  
WHERE (email = emailStudente) AND (AnnoImm > 1999)  
GROUP BY email
```

B)

```
CREATE VIEW numerolscrittiPerCorso (CodCorso, NumeroPartec) AS(  
SELECT CodCorso, COUNT(*)  
FROM MAILINGLIST, CORSO  
    WHERE (CodCorso = Codice) AND (Anno = 3)  
    GROUP BY CodCorso  
)
```

```
SELECT Codice  
FROM CORSO, numerolscrittiPerCorso  
WHERE (CORSO.Codice = numerolscrittiPerCorso.CodCorso) AND  
NumeroPartec = (SELECT (MAX) FROM numerolscrittiPerCorso)
```

C)

```
SELECT email, nome, cognome  
FROM STUDENTE AS S JOIN MAILINGLIST AS M1 JOIN MAILINGLIST AS M2 ON (S.email =  
M1.emailstudente) AND (S.email = M2.emailstudente)  
WHERE (M1.CodCorso <> M2.CodCorso)
```

D)

```
CREATE TABLE STUDENTE (
```

Email varchar(256) PRIMARY KEY,
Nome varchar(100),
Cognome varchar(100) NOT NULL,
Annolmm integer,

CHECK (Annolmm > 1980)

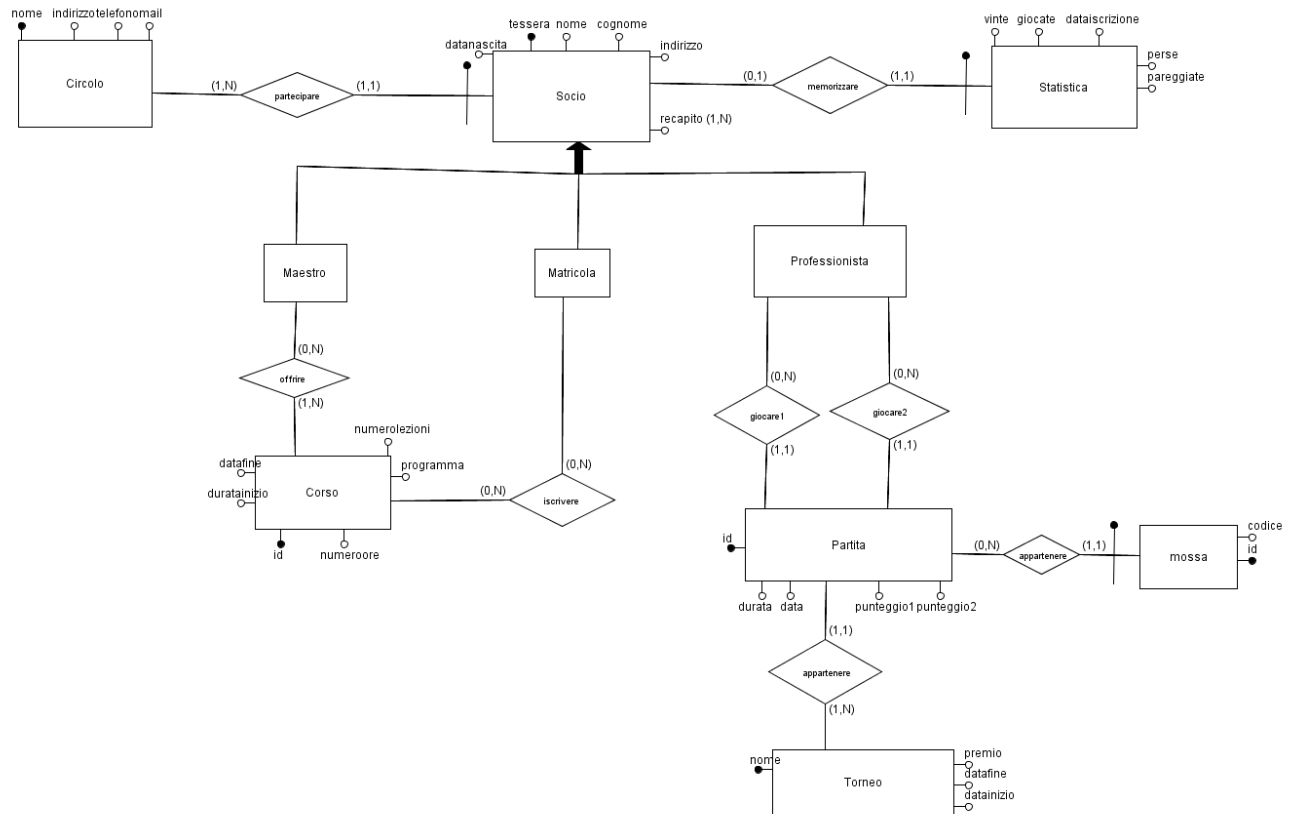
CHECK (COUNT(Nome,Cognome)<=3)

)

E)

Db.STUDENTE.find({\$or:{Nome:Mario},{Annolmm {\$gt:2000}}},{email:1,Cognome:1})

Esercizio 2



Schema Logico

CIRCOLO(nome, indirizzo, telefono, mail)

SOCIO(tessera, nome, cognome, data nascita, nomeCircolo)

STATISTICA(vinto, giocate, pareggiate, perse, data iscrizione, tesseraSocio, nomeCircolo)

MAESTRO(tesseraSocio, nomeCircolo)

MATRICOLA(tesseraSocio, nomeCircolo)

PROFESSIONISTA(tesseraSocio, nomeCircolo)

CORSO(tesseraSocioMaestro, nomeCircolo, id, numeroOre, datainizio, datafine, numerolezioni, programma)

ISCRIZIONE(idCorso, tesseraSocioMatricola, nomeCircolo)

...

B)

Traduzione con Soluzione3

C)

1) 2 * 1 * (2*1)

$$2) 5 * 1 * (2 * (1 + 1))$$

$$3) 1 * 1 * (25 + 25)$$

Esercizio 3

AB → DE

A → C

A → F

A)

{ABF} + F = {ABFDEC} superchiave, controllo se minimale

{AB} + F = {ABDECF} quindi ABF non chiave

FNBC = No

3FN = No

Esercizio 4

A)

Le proprietà ACID delle transazioni sono proprietà che un DMBS deve garantire (e che quindi devono essere rispettate da tutte le transazioni).

A = Atomicità, le transazioni vanno fatte o tutto o niente

C = Consistenza, le transazioni devono lasciare il DB in uno stato consistente rispettando vincoli

I = Isolamento, le transazioni devono essere indipendenti le une dalle altre

D = Persistenza, le transazioni che effettuano commit devono apportare modifiche al DB

Esame Febbraio 2016

Esercizio 1

A)

```
SELECT COUNT(*)  
FROM PLAYLIST  
WHERE CittaUtente = Bologna AND Nome IN  
(SELECT NomePlay  
FROM COMPOSIZIONE  
WHERE TitoloC ="Piazza Grande")
```

B)

```
CREATE VIEW numeroCanzoniREMperPlaylist (NomePlaylist, numero) AS (  
SELECT NomePlay, COUNT(*)  
FROM COMPOSIZIONE JOIN CANZONE  
WHERE Artista = "REM"  
GROUP BY NomePlay  
)
```

```
SELECT Nome, NomeUtente  
FROM PLAYLIST JOIN numeroCanzoniREMperPlaylist on (NomePlaylist = Nome)  
WHERE numero = (SELECT MAX(numero) FROM numeroCanzoniREMperPlaylist)
```

C)

```
SELECT NomePlay  
FROM COMPOSIZIONE  
WHERE TitoloC =  
ALL(SELECT Titolo FROM CANZONE WHERE Artista = Beatles OR Artista = Rolling Stones)
```

D)

```
CREATE TABLE PLAYLIST(  
  Nome varchar(20) PRIMARY KEY,  
  NomeUtente varchar(100) NOT NULL,  
  CittaUtente varchar(100),
```

```
  CHECK(Nome LIKE 'PL_%000'),
```

```
)
```

```
CREATE ASSERTION controllamax100canzoni(CHECK(COUNT(NomePlay,Titoloc)<=100))
```

E)

```
Db.PLAYLIST.find({Nome: "Rock3"},{NomeUtente:1})
```

Esercizio 2

A)

ER

B)

vincoli

C)

$$C(1) = 20 * 1 * (2^2)$$

$$C(2) = 3 * 1 * (20+20)$$

$$C(3) = 5 * 0.5 * (2*(1+1)*2)$$

Esercizio 3

A)

$\{AC\}+F = \{A,C,B,E,D\}$ superchiave, controllo se minimale

$\{A\}+F = \{A\}$

$\{C\}+F = \{C,B,E,A,D\}$ chiave

B)

FNBC: no perché non in 3FN

3FN: per ogni dipendenza $X \rightarrow Y$, X superchiave o Y parte della chiave. No perché $BE \rightarrow AD$
Non rispetta le condizioni

$C \rightarrow B$

$BE \rightarrow AD$

$ABC \rightarrow D$

$C \rightarrow E$

$C \rightarrow BE$

$BE \rightarrow AD$

$AC \rightarrow D$

$T1(\underline{C}, B, E)$

$T2(\underline{B}, E, A, D)$

Esercizio 4

A)

Il teorema CAP afferma che un sistema distribuito può godere al massimo solo di due delle 3 proprietà seguenti

- Availability: sistema/servizio sempre disponibile
- Consistency: tutti i nodi vedono gli stessi dati
- Partition Tolerance: se uno o più nodi vanno offline il servizio continua a funzionare

B)

RTM = 4, WTM = 6

R9 sì RTM = 9

R5 no

R8 sì

W15 sì WTM = 15

R16 sì RTM = 16

R14 no

W11 no

W12 no

R18 si $RTM = 18$

R13 no

Esame2 (Esame 2)

Esercizio 1

A)

Colonne: 9 [4 di UTENTE + 5 di CORSA]

Righe: minimo 0 [se il codice 12 non esistesse]

Massimo 200 se tutte le corse sono associate al codice 12

B)

```
SELECT Codice, Nome, Cognome,  
FROM UTENTE  
WHERE EXISTS (  
    SELECT *  
    FROM CORSA  
    WHERE (costo>100) AND UTENTE.Codice = CORSA.CodUtente)
```

C)

```
SELECT Codice, SUM(Corsa.Costo)  
FROM UTENTE, CORSA  
WHERE ( CORSA.citta = "BOLOGNA"  
AND UTENTE.Codice = Corsa.Codice  
    )  
GROUP BY Codice  
HAVING (COUNT(*)>=3)
```

D)

```
SELECT Nome, Cognome  
FROM UTENTE JOIN CORSA ON UTENTE.Codice = CORSA.CodiceUtente  
WHERE (  
    (et  <30 AND CORSA.citta = BOLOGNA )
```

OR

(et <30 AND CORSA.citta = FIRENZE)

E)

SELECT Numero, Modello

FROM CORSE, TAXI

WHERE (

CORSE.NumeroTaxi =TAXI.Numero

AND Annolmmatricolazione <2000

)

GROUP BY Numero

HAVING(MAX(COUNT(*)))????????????????????????????????

SELECT MAX(NUM), Numero, Modello

FROM TAXI, CONTARECORSE

WHERE(

Annolmmatricolazione <2000 _____

CONTARECORSE. NumeroDelTaxi = TAXI.Numero

)

CREATE VIEW CONTARECORSE(NumeroDelTaxi, NUM) AS(

SELECT NumTaxi AS NumeroDelTaxi, COUNT(*) AS NUM

FROM CORSE

GROUP BY NumTaxi

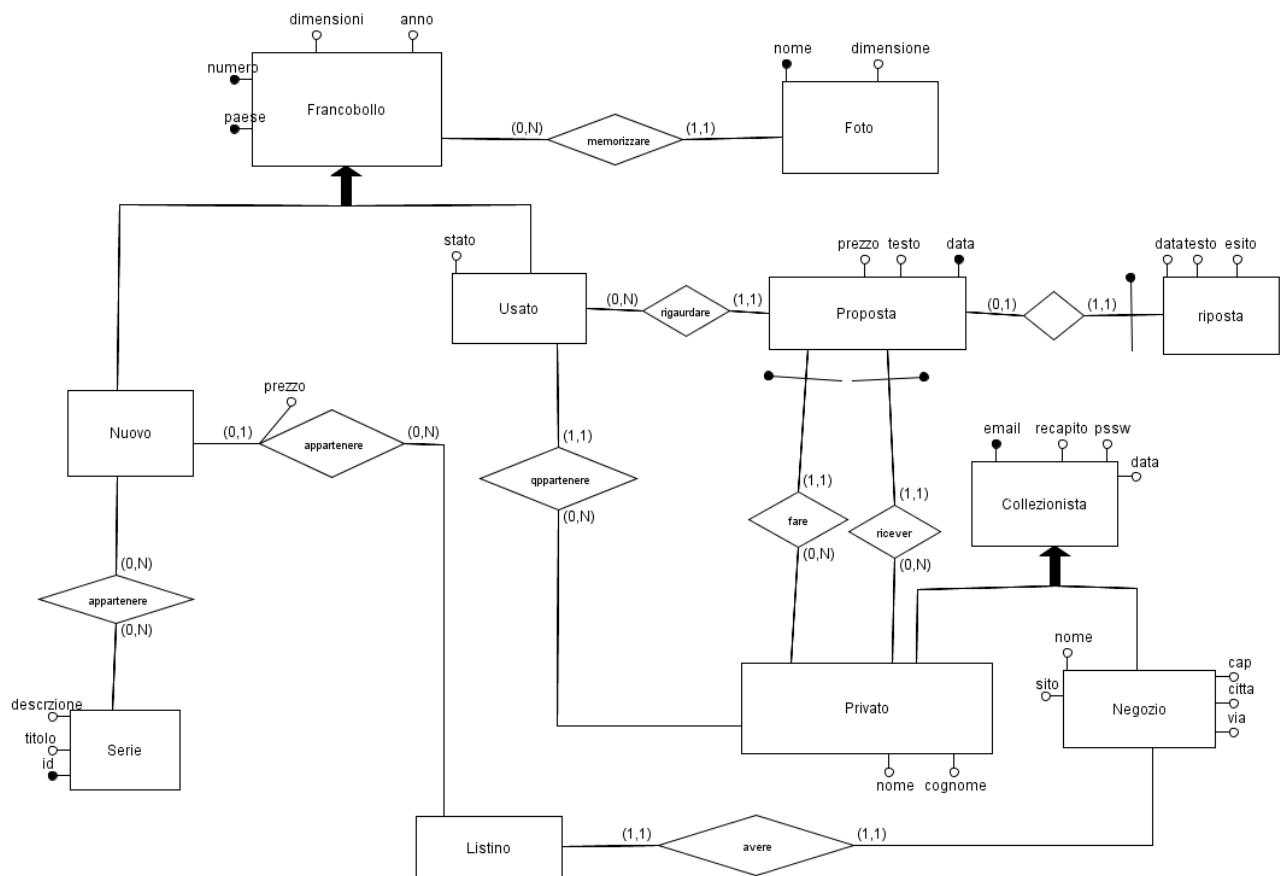
)

F)

Db.TAXI.find({\$or[{"ModelloVeicolo":Toyota Prius"} , {Annolmmatricolazione:{\$gt:2015}},
{Numero:1 , _id:0}

Esercizio 2

A)



B)

Soluzione 3

C)

$$1 * 1 * (2 * (1 + 1 + 1))$$

$$3 * 0.5 * (2 * (5 + 5))$$

$$2 * 1 * (3 * (1 + 1))$$

Esercizio 3

Codice Durata Controindicazioni

Tipologia ModoAssunzione

NomeFarmaco Quantit Tipologia

NomeFarmaco Tipologia Costo

Codice Durata

A)

{NomeFarmaco, Tipologia, Codice}+F

= {NomeFarmaco, Tipologia, Codice, Costo, Durata, Quantit , ModoAssunzione, Controindicazioni}

Superchiavi: {NomeFarmaco, Tipologia, Codice}

{NomeFarmaco, Tipologia, Codice, Costo}

B)

NON il FNBC (per ogni dipendenza $X \rightarrow Y$, X superchiave)

NON in 3FN (per ogni dipendenza $X \rightarrow Y$, X superchiave o Y parte della chiave)

Decomposizione:

Codice Controindicazioni, Durata

Tipologia ModoAssunzione

NomeFarmaco Quantit Tipologia Costo

T1(Codice, Controindicazioni, durata)

T2(Tipologia, ModoAssunzione)

T3(NomeFarmaco, Quantit Tipologia Costo)

T4(Codice, NomeFarmaco)_

Esercizio 4

A)

Data Querying fa riferimento a tutte le operazioni effettuate su un database (relazionale o meno) per mostrare e/o filtrare determinati dati presenti nel database.

Data Mining fa riferimento a tutte le operazioni effettuate su un database in modo da riuscire ad estrapolare conoscenze ulteriori, come correlazioni tra i dati, non “esplicitamente” presenti nel database.

B)

1) Lo schedule rispetta il 2PL perch  entrambe le transazioni non acquisiscono altri lock dopo aver rilasciato il primo. (La fase di acquisizione   sempre prima della fase di rilascio)

2) Lo schedule non rispetta il S2PL perch  i lock non sono rilasciati dopo la commit

Esame 30 giugno 2016

Esercizio 1

A)

```
SELECT T.Nome, T.Citta
FROM TEATRO AS T
WHERE T.Capienza =
      (SELECT COUNT(*)
       FROM ABBONAMENTO AS A
       WHERE (T.Nome = A.NomeTeatro))
```

B)

```
SELECT U.Codice, U.Nome, U.Cognome
FROM UTENTE AS U JOIN ABBONAMENTO AS A1 JOIN ABBONAMENTO AS A2 ON
U.Codice = A1.CodUtente AND U.Codice = A2.CodUtente
WHERE Eta > 50 AND A1.ID <> A2.ID
```

Alternativa

```
SELECT U.Codice, U.Nome, U.Cognome
FROM UTENTE AS U, ABBONAMENTO AS A1, ABBONAMENTO AS A2
WHERE (U.Codice = A1.CodUtente) AND (U.Codice = A2.CodUtente) AND (eta>50) AND
(A1.id <> A2.id)
```

C)

```
CREATE VIEW numeroAbbonamentiPerCitta (Citta, somma) AS (
SELECT citta, COUNT(*)
FROM TEATRO, UTENTE, ABBONAMENTO
WHERE (intref) AND (eta BETWEEN 18 AND 25)
      GROUP BY Citta
)
```

```
SELECT Citta
```


FROM numeroAbbonamentiPerCitta

WHERE somma = (SELECT MAX(somma) FROM numeroAbbonamentiPerCitta)

D)

CREATE TABLE ABBONAMENTO (

 ID char(15) PRIMARY KEY,

 CodUtente varchar(20),

 NomeTeatro varchar(100),

 Costo integer,

 UNIQUE (CodUtente, NomeTeatro),

 CHECK (ID LIKE "N_%00"),

 FOREIGN KEY CodUtente REFERENCES UTENTE(Codice)

 FOREIGN KEY NomeTeatro REFERENCES TEATRO(Nome)

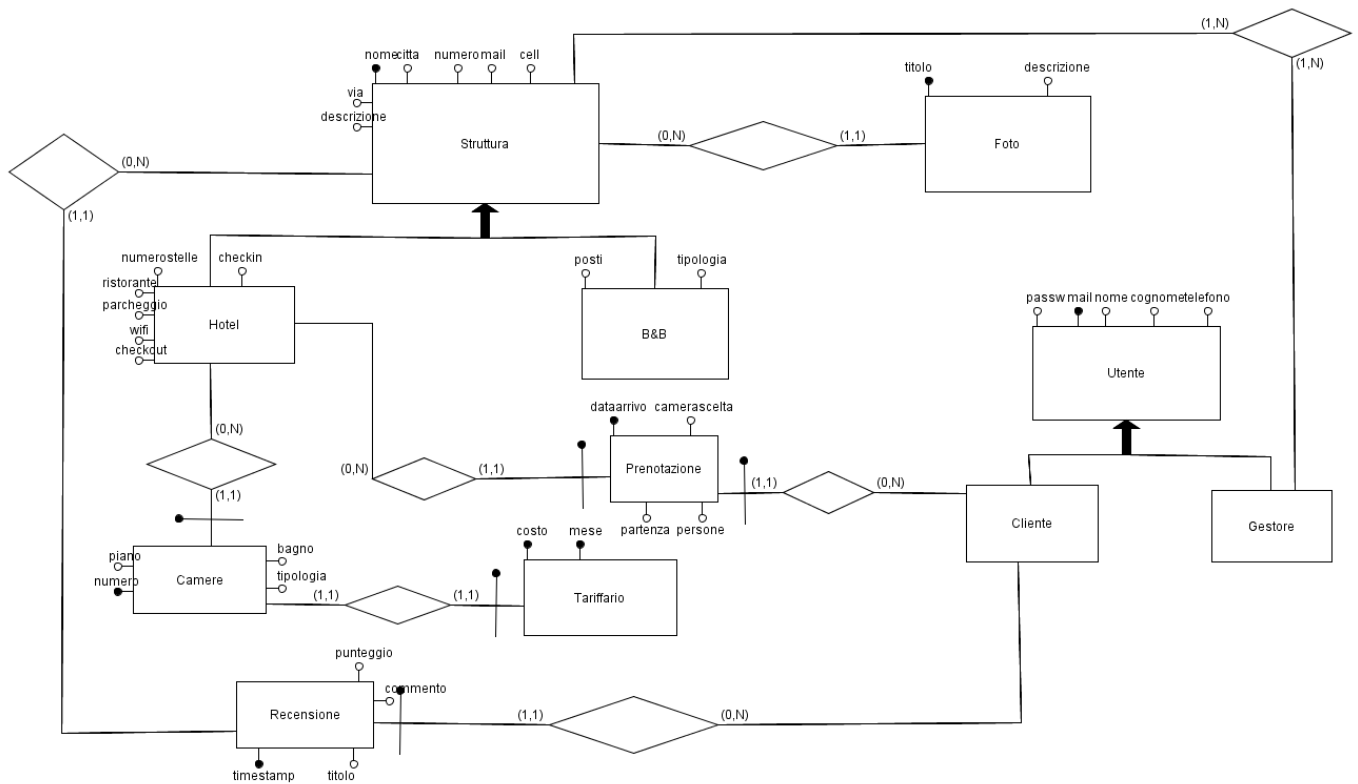
)ENGINE = INNODB

E)

Db.UTENTE.find({Nome:"Mario", eta:{\$gt:30}}, {Codice:1, Cognome:1, _id:0})

Esercizio 2

A)



B)

Vincoli

C)

Frequenza * coefficiente * (scritture * peso + letture)

$$C(1) = 1 * 0.5 * (2 * 3 * 20)$$

$$C(2) = 4 * 1 * (2 * (1+3))$$

$$C(3) = 2 * 1 * ((2)*50)$$

Esercizio 3

BD->C

AB->D

FC->E

{ABFC}+F = {ABFCDE} superchiave

{ABF}+F = {ABDCFE} chiave

FNBC = no per tutte le dipendenze

3FN = no per tutte le dipendenze

Esercizio 4

a)

CAP Theorem Gi scritto mille volte sopra

b)

WTM = 2 RTM = 5

W7 = si WTM = 7

R6 = no

R12 = si RTM = 12

R10 = si

W11 =no

R13 =si RTM =13

W21 = si WTM = 21

R20 = no

W18 = no

Esame senza nome

A)

Min 0 max 10

B)

```
SELECT AVG(eta)
FROM UTENTE
WHERE CF IN (SELECT CFUTENTE FROM ABBONAMENTO WHERE COSTO =100)
```

C)

```
SELECT CF, Nome, Cognome
FROM UTENTE
WHERE CF NOT IN
(SELECT CFUtente
    FROM ABBONAMENTO
    WHERE NomeRivista = ComputerWorld)
AND CF IN
(SELECT CFUtente
    FROM ABBONAMENTO
    WHERE NomeRivista = ScienzaOggi)
```

D)

```
CREATE VIEW numeroUtentiPerRivista (nomeRivista, num) AS (
SELECT nomeRivista, COUNT(*)
FROM ABBONAMENTO AS A JOIN UTENTE As U ON (A.Cfutente = U.Cf)
WHERE eta < 30
GROUP BY nomeRivista
)
```

```
SELECT nomeRivista
FROM RIVISTA JOIN numeroUtentiPerRivista AS N
ON (R.nomeRivista = N.nomeRivista)
WHERE Tipologia = Mensile AND N.num =
      (SELECT MAX (num) from numeroUtentiPerRivista)
```

E)

```
SELECT U.citta
FROM UTENTE AS U JOIN ABBONAMENTO AS A ON (U.Cf = A.CfUtente)
WHERE EXISTS (5000<=
      SELECT SUM(A.Costo)
      FROM ABBONAMENTO AS A JOIN UTENTE AS U ON (A.cfutente = U.cf)
      WHERE A.NomeRivista = ScienzaOggi
      GROUP BY (U.citta)
    )
```

F)

```
CREATE PROCEDURE procedura (IN cf varchar(30))
BEGIN
DECLARE A INT DEFAULT 0;
A = (SELECT COUNT(*) FROM ABBONAMENTO WHERE Cfutente = cf)
IF A = 0
THEN
DELETE(*) FROM ABBONAMENTO AS A JOIN RIVISTA AS R ON
(A.NomeRivista = R.nomerivista) WHERE Tipologia = Mensile
END IF
END
```

Esame 10 gennaio 2017

Esercizio 1

A)

```
SELECT DISTINCT CODICE,COGNOME
FROM UTENTE
WHERE PROFESSIONE = IMPIEGATO AND EXISTS(
    SELECT *
    FROM POLIZZA
    WHERE NOMESOC = GENERALI)
```

B)

```
SELECT Nome, Citta, Telefono
FROM SOCIETA
WHERE Sede = BOLOGNA AND EXISTS(
    500000<=
    (SELECT SUM(Importo)
    FROM POLIZZA
    GROUP BY NomeSoc
    ))
```

C)

```
SELECT Codice,Cognome
FROM UTENTE JOIN POLIZZA ON (Codice = CodUtente)
WHERE Importo =
(SELECT MAX(IMPORTO)
FROM POLIZZA
WHERE NomeSoc = Generali)
```

D)

```
SELECT Codice,Cognome
FROM UTENTE
WHERE Codice IN (
    SELECT CodUtente
    FROM POLIZZA)
AND Codice NOT IN (
    SELECT CodUtente
    FROM POLIZZA
    WHERE NomeSoc = Generali
    )
```

E)

```
Db.POLIZZA.find({$or:{NomeSoc:Generali,importo:{$gt:500000}}},{id:1})
```

Esame 9 Giugno 2016

Esercizio 1

A)

```
SELECT COUNT(*)
FROM PACCO
WHERE tipologia = Prioritario AND Codice IN
SELECT CodicePacco
FROM GIACENZA
WHERE idUfficio IN
SELECT id
FROM UFFICIO
WHERE NumeroDipendenti>5 AND Citta=Bologna
```

B)

```
SELECT id
FROM UFFICIO
WHERE Citta = BOLOGNA and id NOT IN
SELECT idUfficio
FROM GIACENZA, PACCO
WHERE CodicePacco = Codice AND Tipologia = Prioritario
```

C)

```
CREATE VIEW mediaGiorniPerUfficio (idUfficio, media) AS (
SELECT idUfficio, AVG(NumGiorni)
FROM GIACENZA
GROUP BY idUfficio
)

SELECT idUfficio
```



```
FROM mediaGiorniPerUfficio
WHERE media = (SELECT MIN(media) FROM mediaGiorniPerUfficio)
```

D)

```
CREATE TABLE PACCO (
Codice varchar(20) PRIMARY KEY,
Tipologia varchar(15) DEFAULT "Generico",
Peso integer,

CHECK(Peso<20),
CHECK(Codice LIKE 'COD%')
)
```

E)

```
Db.PACCO.find({Tipologia:Prioritario, peso{$gt:5}},{codice:1})
```

Esercizio 4

A)

Un sistema distribuito pu avere solamente 2 delle 3 propriet elencate sotto:

- Consistency: I dati sono consistenti
- Availability: il sistema sempre disponibile
- Partition Tolerance: il sistema resta funzionante anche se uno o pi nodi si staccano dalla rete

B)

Wlock(y)	
Rlock(x)	
W(y)	
R(x)	
Unlock(x)	
Unlock(y)	
	Wlock(z)
	W(z)
	Unlock(z)
	commit
Wlock(z)	

W(z)	
Unlock(z)	
commit	

Rispetta il 2PL? NO

Perch T0 dopo aver rilasciato i primi due lock, ne acquisisce un altro

Rispetta il S2PL? NO

I lock devono essere rilasciati tutti dopo la commit

Esame pi recente appello 2021

Esercizio 1

A)

[min:0 max:10]

B)

```
CREATE VIEW righeComplessivePerSoftware (NomeSoftware, TOT) As (  
SELECT NomeSoftware, SUM(NumeroRigheCodice) AS TOT  
FROM SVILUPPO  
GROUP BY NomeSoftware  
)
```

```
SELECT Licenza, NomeCreatore  
FROM SVILUPPO AS S JOIN righeComplessivePerSoftware AS R  
ON (S.NomeSoftware = R.NomeSoftware)  
WHERE R.TOT = (SELECT MAX(TOT) FROM righeComplessivePerSoftware)
```

C)

```
SELECT Nome  
FROM SOFTWARE  
WHERE Nome IN  
(SELECT NomeSoftware  
FROM SVILUPPO  
WHERE NomeLinguaggio = "Java")  
AND Nome IN  
(SELECT NomeSoftware  
FROM SVILUPPO  
WHERE NomeLinguaggio = "Javascript")
```

D)

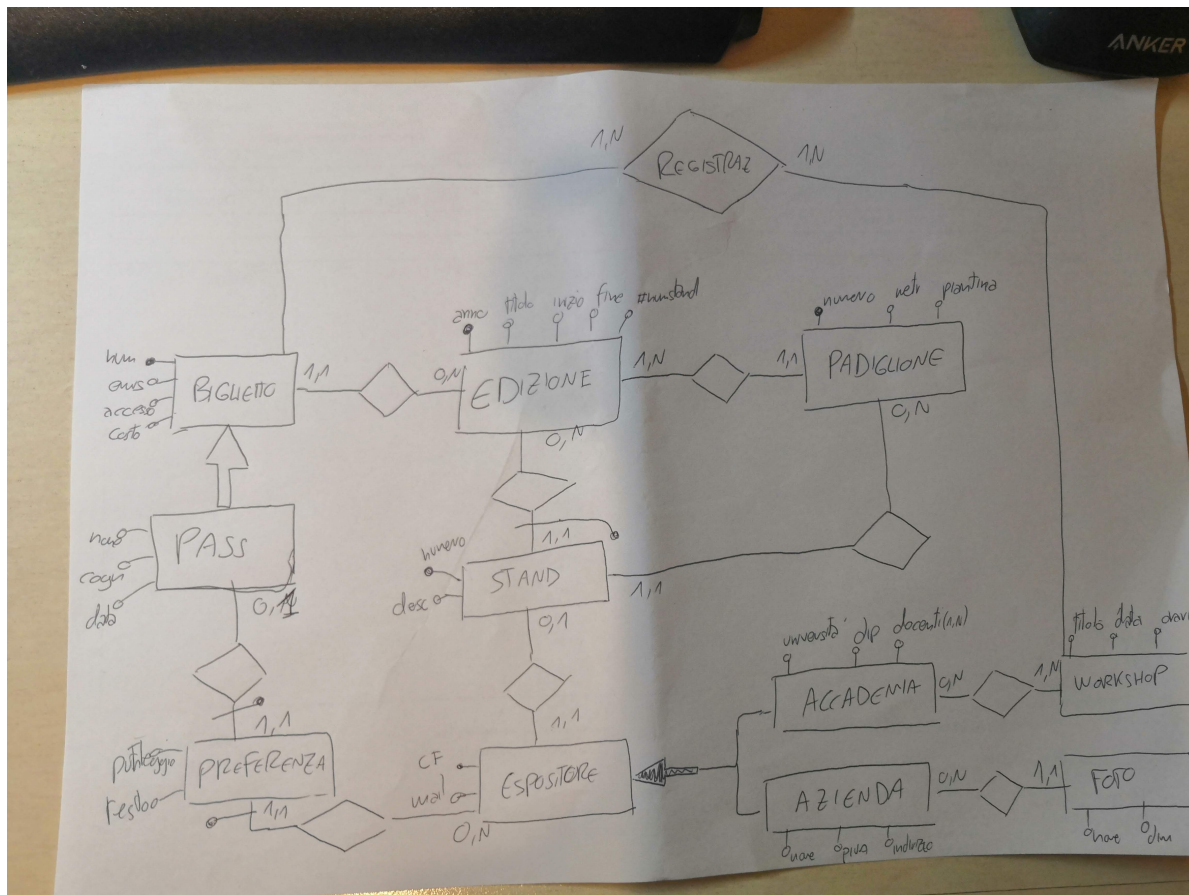
```
SELECT Nome
```

```
FROM SOFTWARE JOIN SVILUPPO ON (Nome = NomeSoftware)
WHERE (Licenza = "GPL")
GROUP BY Nome
HAVING (COUNT(*)>=2)
```

```
SELECT Nome
FROM SOFTWARE AS S JOIN SVILUPPO AS S1 JOIN SVILUPPO AS S2
ON (S.Nome = S1.NomeSoftware) AND (S.Nome = S2.NomeSoftware)
WHERE S1.NomeLinguaggio <> S2.NomeLinguaggio AND Licenza = 'GPL'
```

Esercizio 2

ER provvisorio



Soluzione che minimizza le tabelle: Soluzione 1 quindi nelle generalizzazioni accorpare le entità figlie nell'entità padre e unire al padre tutti gli attributi + un campo TIPO

Analisi della ridondanza:

$$C(1 \text{ rid}) = 2 * 0.5 * (1) = 1$$

$$C(2 \text{ rid}) = 1 * 1 * (5 * (20+20)) = 200$$

$$C(1) = 2 * 0.5 * (20+20) = 40$$

$$C(2) = 1 * 1 * (5 * (20+20)) = 200$$

Calcolo speedup con la formula $c(S)/c(R \text{ rid})$

$$= 240/201 = 1.2$$

Quindi la ridondanza migliora le prestazioni di un fattore di circa 1.2

Calcolo occupazione di memoria

#numerostad un integer, ci sono 5 edizioni quindi $4B * 5 = 20\text{Bytes}$

Conclusione

L'occupazione di memoria non è elevata e si potrebbe scegliere di mantenere la ridondanza per migliorare le prestazioni di un fattore di 1.2

Oppure

Il fattore di miglioramento delle prestazioni è inferiore a 2, pertanto non così elevato e si potrebbe scegliere di rimuovere la ridondanza

Esercizio 3

Provo con

$\{\text{NomeSquadra Anno NumeroTappa}\} + F =$

$\{\text{NomeSquadra, Anno, NumeroTappa, NumeroComponenti, CittaPartenza, CittaArrivo, Edizione, NomeDirettore, Regione}\}$

una superchiave e anche chiave perché è minimale

Un'altra superchiave quindi può essere

$\{\text{NomeSquadra Anno NumeroTappa}\} +$ un altro attributo qualsiasi

2FN: No, ci sono dipendenze parziali

3FN: per ogni $X \rightarrow Y$ X è superchiave o Y è parte della chiave

(la 1,2,3,4,5,6 non vanno bene)

FNBC: No per 3FN

NomeSquadra Anno \rightarrow NumeroComponenti

Anno NumeroTappa \rightarrow CittaPartenza CittaArrivo

CittaPartenza \rightarrow Regione

Anno \rightarrow Edizione

NomeSquadra Anno \rightarrow NomeDirettore

Normalizzazione:

NomeSquadra Anno -> NomeDirettore NumeroComponenti

Anno NumeroTappa -> Cittapartenza cittarrivo

Cittapartenza -> regione

Anno -> Edizione

T1 (NomeSquadra Anno, NomeDirettore NumeroComponenti)

T2 (Anno NumeroTappa , Cittapartenza cittarrivo)

T3 (Cittapartenza ,regione)

T4 (Anno ,Edizione)

Ripasso.pdf

ES1 normalizzazione

Dipendenze funzionali:

1) $BC \rightarrow A$

$A \rightarrow C$

Chiave $\{B, C\}$

FNBC: $X \rightarrow Y$ X superchiave = la 2 no

3FN: $X \rightarrow Y$ superchiave o Y sottoinsieme della chiave = si

2FN: non ha dipendenze parziali = non ci sono quindi si

2)

$\{CDB\} + F = \{C, D, B, E, A\}$ chiave

$\{CD\} + F = \{C, D, E, A\}$

$\{DB\} + F = \{D, B, A\}$

$\{CB\} + F = \{C, B, A\}$

ES2 QUERY SQL

1)

```
SELECT AVG(LISTINO.Prezzo)
FROM LISTINO, PRODOTTI, NEGOZI
WHERE (PRODOTTI.Codice = LISTINO.CodProdotto) AND
(NEGOZI.Id = LISTINO.IdNegozio) AND
(PRODOTTI.TipoProdotto = LAVATRICE) AND
(NEGOZIO.Citta = BOLOGNA)
```

2)

```
CREATE VIEW PrezzoMinimoPerLavatrice (CodiceProdotto, PrezzoLav) AS (
SELECT CodiceProdotto, MIN(Prezzo) AS PrezzoLav
FROM LISTINO JOIN PRODOTTO ON (L.CodiceProdotto = P.Codice)
WHERE P.TipoProdotto = Lavatrice)
```


GROUP BY CodiceProdotto

)

```
SELECT CodiceProdotto, Nome, Id
FROM NEGOZIO JOIN LISTINO JOIN PrezzoMinimoPerLavatrice AS P ON
(N.Id = L.IdNegozio) AND (P.CodiceProdotto = L.CodiceProdotto)
WHERE L.Prezzo = (SELECT PrezzoLav FROM PrezzoMinimoPerLavatrice)
```

3)

```
SELECT PRODOTTI.Marca
FROM PRODOTTI
WHERE (TipoProdotto = Lavatrice) AND Codice IN
    (SELECT CodiceProdotto
     FROM LISTINO
     WHERE IdNegozio IN(
         SELECT Id
         FROM NEGOZI
         WHERE (Citta = BOLOGNA)))
```

4)

```
SELECT Nome
FROM NEGOZIO
WHERE Citta = "Bologna" AND Id IN (
    SELECT IdNegozio
    FROM LISTINO
    GROUP BY IdNegozio
    HAVING COUNT(*)>=2)
```

5)

```
SELECT N.Nome
```

```

FROM NEGOZI AS N JOIN PRODOTTI AS P JOIN LISTINO AS L
ON (id = IdNegozio) AND (Codice = CodiceProdotto)
WHERE Tipoprodotto = Lavatrice AND Citta = Bologna AND Marca = Ariston AND Prezzo >
(SELECT AVG(Prezzo)*1.20
FROM FROM NEGOZI AS N JOIN PRODOTTI AS P JOIN LISTINO AS L
ON (id = IdNegozio) AND (Codice = CodiceProdotto)
WHERE Tipoprodotto = Lavatrice AND Citta = Bologna AND Marca = Ariston
)

```

ES3 transazioni

Parte 1)

R_lock0(x)	
R0(x)	
W_lock0(y)	
W0(y)	
	W_lock1(y)
Unlock0(x)	
Unlock0(y)	
	W1(y)
	Unlock1(y)
	A(T1)
C(T0)	

Non rispetta lo S2PL perch i lock sono rilasciati prima di completare la transazione.

Rispetta invece il 2PL, perch entrnabe le transazioni, dopo aver rilasciato i lock, non ne acquisiscono altri.

Parte 2)

RTM(X)=10

WTM(X)=7

1. T₁₅: r₁₅(x)= consentita, RTM = 15
2. T₈: r₈(x)= consentita, RTM = 15
3. T₆: r₆(x)= abortita
4. T₁₈: w₁₈(x)= consentita, WTM = 18

5. T_{16} : $w_{16}(x)$ = abortita

6. T_{20} : $r_{20}(x)$ = consentita, $RTM = 20$

$Rt(x) = t < WTM$ la transazione uccisa, $t \geq WTM$ la transazione consentita e RTM aggiornato con il massimo tra il suo valore precedente e il valore di t .

$Wt(x) = t < WTM$ oppure $t < RTM$ la transazione uccisa altrimenti viene consentita e WTM viene aggiornato con il valore di t .

Parte 3)

CK = Checkpoint

C = transazione completata

B = transazione iniziata

A = transazione abortita

D/I/U = delete, insert, update

$CK(T_1, T_2, T_3)$

$UNDO = \{T_2, T_4\}$

$REDO = \{T_1, T_3\}$

ES4 cardinalit

$R1(\underline{A}, B, C, D)$

$R2(\underline{E}, F, G)$

$R1 = 25$ righe

$R2 = 10$ righe

$R2.F \rightarrow R1.A$

Indicare cardinalit delle seguenti query

1) colonne: 4 (A,B,C,D)

Righe: min 0 (caso in cui non ci fosse A = 20), max 1

2) colonne: 4 (A,B,C,D)

Righe: min 0, max 25

3) colonne: 7

Righe: min 0, max 250 (25*10)

4) colonne: 7

Righe: minimo 0 (se f sono tutti NULL), massimo 10 (quelle della tabella più piccola)

5)colonne: 7

Righe: 10

6) colonne: 2 (A + quella del count)

Righe: minimo 1 (tutti gli elementi di f sono uguali e quindi c'è un solo raggruppamento con il count a 10)

Massimo 10 (tutti diversi quindi in pratica non raggruppa e tutti i count sono a 1)

7) colonne: 4 (quelle di A)

Righe: 0 se g non ha mai 15

Righe: 25 se esiste almeno un g che sia 15 perché stampa tutte le righe della R1

8) colonne: 2

Righe: minimo 0 se ogni f è diverso quindi saltano fuori 10 raggruppamenti con count = 1 che non rispettano l'having

Massimo: 5 righe ovvero se tutti i raggruppamenti hanno count = 2

Ripasso2.pdf

ES1

1) G->CF

2) G->BD

3) $BD \rightarrow A$

4) $BCG \rightarrow E$

A)

$\{BCG\} + F = \{B, C, G, E, C, D, A\}$ superchiave, controllo se minimale

$\{BG\} + F = \{B, G, C, B, D, A, E\}$

$\{G\} + F = \{G, C, F, B, D, A, E\}$ chiave

B)

FNCB: per ogni dipendenza $X \rightarrow Y$, X superchiave NO

3FN: per ogni dipendenza $x \rightarrow Y$, X superchiave o Y parte della chiave NO

2FN: se non ci sono dipendenze parziali, SI

C)

Come normalizzare questa tabella?

$T1(G, C, F, B, D, E)$

$T2(B, D, A)$

ES2

A)

Una chiave della tabella $\{\text{NumEdizione}, \text{Canzone}\}$

$\{\text{NumEdizione}, \text{Canzone}\} + F =$

$\{\text{NumEdizione}, \text{Canzone}, \text{Sede}, \text{Teatro}, \text{MaxP}, \text{Autore}, \text{Durata}\}$

Una superchiave qualsiasi $\{\text{NumEdizione}, \text{Canzone}, \text{altroAttributo}\}$

B)

FNBC: NO

3FN: NO

2FN: NO ($\text{NumEdizione} \rightarrow \text{Sede}$ una dipendenza parziale)

C)

Come normalizzare la tabella?

Le regole sono già indipendenti tra loro, ma alcune si possono semplificare

$\text{NumEdizione} \text{ ~~Sede~~$

NumEdizione -> Sede

T1(NumEdizione, Sede, MaxP)

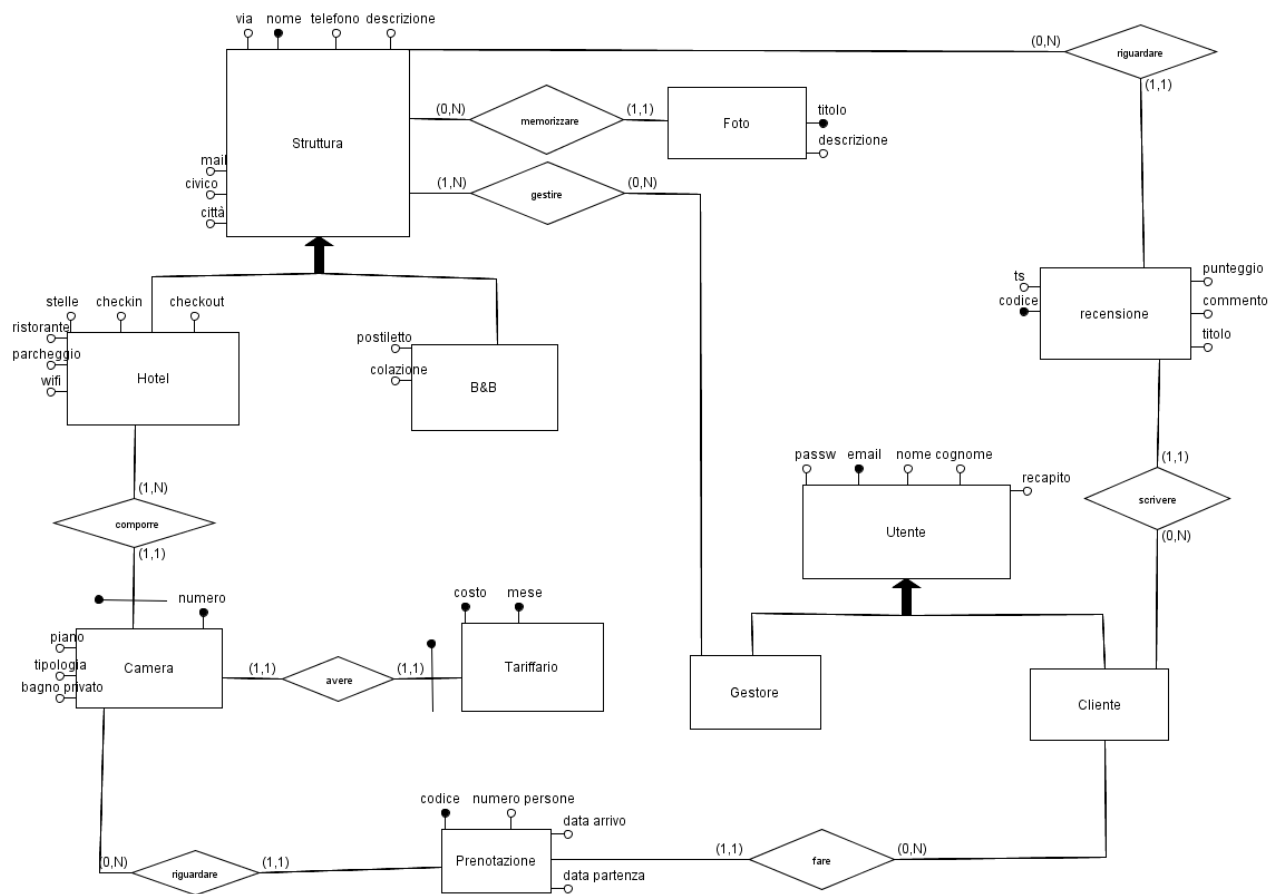
Sede Teatro

T2(Sede, Teatro)

NumEdizione Canzone -> Autore Durata

T3(NumEdizione Canzone Autore Durata)

Esercizio Progettazione



```
CREATE VIEW totalerighecodiceperprogetto (NomeSoftware, TOT) AS(  
SELECT NomeSoftware, SUM(numeroRighe)  
FROM SVILUPPO AS S,  
GROUP BY NomeSoftware  
)
```

```
SELECT Licenza, NomeCreatore  
FROM SOFTWARE AS S JOIN totalerighecodiceperprogetto AS T  
ON (S.NomeSoftware = T.NomeSoftware)  
WHERE TOT = (SELECT MAX(TOT) )
```

```
SELECT Nome  
FROM CLIENTI  
WHERE NrPatente NOT IN  
      (SELECT PatenteCliente  
      FROM NOLEGGIO  
      )
```

```
SELECT Nome
FROM CLIENTI
WHERE NOT EXISTS
(SELECT *
    FROM NOLEGGIO
    WHERE (CLIENTI.NrPatente = NOLEGGIO.PatenteCliente))
```