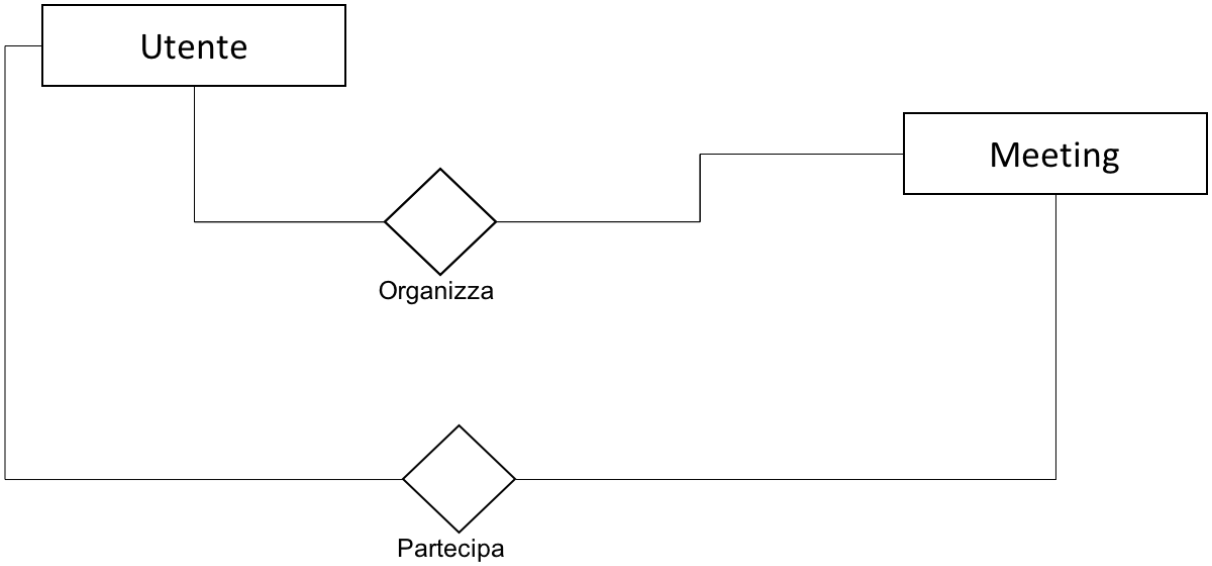
Basi di Dati -- A.A. 2014-2015
Prof. Alfredo Pulvirenti
Esame 03 luglio 2015

A)
Si vuole progettare un database per la gestione dei meeting. Il sistema consente di organizzare dei meeting remoti tramite un meccanismo ad inviti. Un utente (leader) inizia un meeting proponendo una serie di date e di ore. Allo stesso tempo invita un insieme di utenti (membri) a partecipare al meeting. Questi entro 7 giorni dalla creazione del meeting devono dare le preferenze le quali verranno notificate al leader. Una volta ricevute tutte le preferenze o superati i sette giorni il leader identifica la data-ora più condivisa e rende effettivo il meeting notificandolo ai membri.
Dopo il meeting il leader elegge uno dei membri per stilare una minuta contenente tutti i punti trattati nel meeting. L'utente eletto una volta compilata la minuta effettua una notifica a tutti gli utenti.

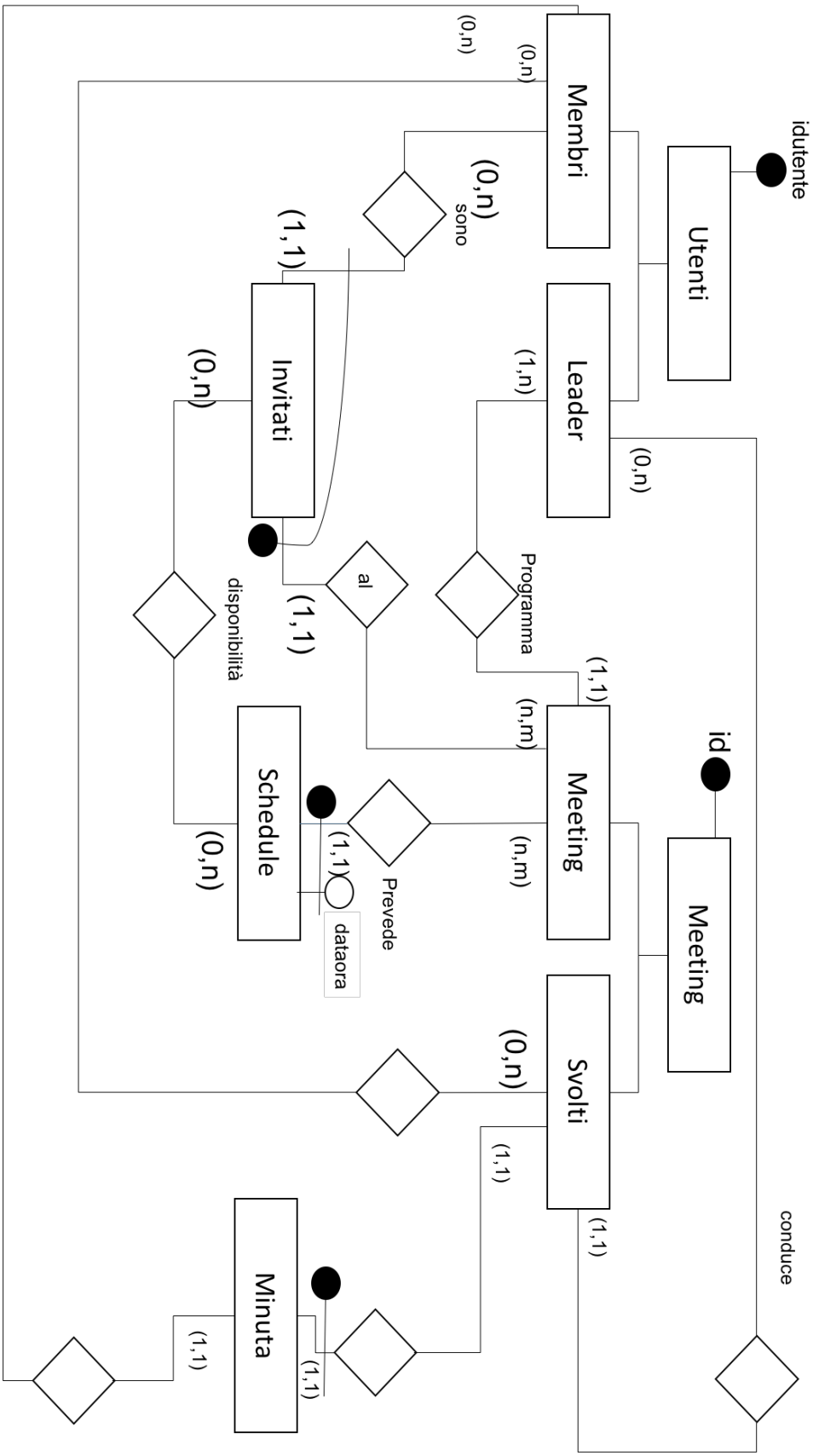
1. Effettuare un'analisi dei requisiti e sviluppare la progettazione concettuale usando la strategia top-down [4 punti];

   Schema Scheletro:



   Applicando le trasformazioni della metodologia di progettazione top-down otteniamo il seguente schema finale. Si noti che sono definite solo le chiavi primarie e le relazioni. Gli altri attributi sono personalizzabili.

Basi di Dati -- A.A. 2014-2015
Prof. Alfredo Pulvirenti
Esame 03 luglio 2015



2. Effettuare quindi la progettazione logica descrivendo tutte le relazioni e dando il codice SQL di alcune delle tabelle identificate includendo eventuali vincoli di integrità sui domini e inter-relazionali [4 punti];

   Lo schema relazionale e' il seguente:

   Utente(**idutente**, nome, cognome,email)
   Meeting(**id**,titolo,idleader,dataapertura_sondaggio,
   data_chiusura_sondaggio, data_ora_meeting)
   Partecipante(**idpartecipante,idmeeting**)
   Sondaggio(**idmeeting,data_ora**)
   Preferenze(**idmeeting,data_ora,idpartecipante**)
   Minuta(**idmeeting**,titolo,testo,idpartecipante_segretario)

Come vincolo non esprimibile abbiamo il valore di data_ora_meeting che viene valorizzato sulla base del contenuto dei record corrispondenti di Preferenze.

La notifica viene gestita attraverso una email agli utenti una volta che il leader ha deciso data e ora.

A seguire due comandi SQL per la creazione delle tabelle utente e Meeting.

```
CREATE TABLE utente (
  id INT,
  nome  VARCHAR(45),
  cognome VARCHAR(45),
  PRIMARY KEY (id))

CREATE TABLE meeting (
  id INT NOT NULL,
  titolo VARCHAR(45) NOT NULL,
  idleader INT NOT NULL,
  data_apertura_sondaggio DATE NOT NULL,
  data_chiusura_sondaggio DATE NOT NULL,
  DataOraMeeting DATETIME NULL,
  PRIMARY KEY (id),
  CONSTRAINT iduser
    FOREIGN KEY (idleader)
    REFERENCES utente id)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

Basi di Dati -- A.A. 2014-2015
Prof. Alfredo Pulvirenti
Esame 03 luglio 2015

3. Discutere eventuali anomalie e indicare se il database ottenuto è in forma normale 3NF o BCNF [3 punti];

   Lo schema non presenta anomalie e tutte le tabelle sono in forma normale. La tabella Meeting e la tabella Minuta hanno una relazione 1 a 1 e potevano essere raggruppate in un'unica tabella.

4. Implementare le seguenti query in algebra relazionale:
   a. Trova gli utenti che hanno partecipato ai meeting il cui leader è "Mario Rossi" ma non ai meeting il cui leader è "Giovanni Gialli" [2 punti];

$$R1 =: \pi_{id}(\sigma_{idutente=idleader}(\pi_{nome=\text{Mario} \wedge cognome="Rossi}(Utenti) \times Meeting))$$

$$R2 \coloneqq \pi_{idpartecipante}(\sigma_{id=idmeeting}(R1 \times Partecipante))$$

$$R3 =: \pi_{id}(\sigma_{idutente=idleader}(\pi_{nome="\text{Giovanni}" \wedge cognome="Gialli"}(Utenti) \times Meeting))$$

$$R4 \coloneqq \pi_{idpartecipante}(\sigma_{id=idmeeting}(R3 \times Partecipante))$$

$$R2 - R4$$

b. Trovare le coppie di utenti che hanno partecipato sempre agli stessi meeting [3 punti];

Utenti coppie che hanno partecipato agli stessi meeting:

$$P1 \leftarrow \delta_{\substack{idpartecipante \rightarrow idp1 \\ idmeeting \rightarrow idm1}}(Partecipante)$$
$$P2 \leftarrow \delta_{\substack{idpartecipante \rightarrow idp2 \\ idmeeting \rightarrow idm2}}(Partecipante)$$

$$R1 \leftarrow \pi_{idp1,idp2,idm1}\,(P1 \bowtie_{\substack{idp1>idp2 \wedge \\ idm1=idm1}} P2)$$
$$R2 \leftarrow \pi_{idp1,idp2,idm1}\,(P1 \times P2)$$

Coppie che non hanno partecipato agli stessi meeting
Quindi esiste un meeting a cui uno dei due ha partecipato e l'altro no e viceversa

$$R3 \leftarrow \pi_{idp1,idp2}(R2 - R1)$$

Togliamo queste coppie e restano solo quelli che hanno partecipato assieme sempre agli stessi meeting

$$\pi_{idp1,idp2}(R1) - R3$$

c. Trovare gli utenti che hanno partecipato a tutti i meeting indetti da "Mario Rossi" [3 punti];

$$R1 = : \delta_{id \rightarrow idmeeting}\pi_{id}(\sigma_{idutente=idleader}(\pi_{nome=\text{Mario} \wedge cognome="Rossi}(Utenti) \times Meeting))$$
$$Partecipanti \div R1$$

5. Implementare le seguenti query in SQL
   a. Trovare i meeting le cui minute contengono la parola "Java" scritte dall'utente "Filippo Verdi" [1 punti];

```
SELECT idmeeting
FROM minuta
WHERE testo like '%Java%'
```

b. Trovare per ogni utente trova il numero di meeting meeting a cui ha partecipato nell'anno 2014 [2 punti];

```
SELECT count(*), idpartecipante
FROM meeting, patecipante
WHERE idmeeting = id and
year(data_ora_meeting)=2014
Group bt idpartecipante
```

c. Trovare le coppie di utenti che hanno partecipato agli stessi meeting o che hanno un utente in comune che ha partecipato allo stesso meeting [4 punti];

```
CREATE VEIEW coppie
AS SELECT p1.idpartecipante as part1 ,p2.idpartecipante  as part2
FROM partecipanti p1, partecipanti p2
WHERE p1.idpartecipante <> p2.idpartecipante
AND p1.idmeeting = p2.idmeeting


SELECT c1.part1, c2.part2
FROM coppie c1, coppie c2
WHERE c1 .part2= c2.part1
UNION
SELECT * FROM coppie
```

B)
Descrivere le proprietà ACIDE delle transazioni; La gestione dei lock con particolare attenzione al 2PL e il 2PL stretto [4 punti].

Vedere descrizione soluzione da libro di testo o da slide.