Soluzione prova di Informatica
Indirizzo Abacus – esame 2008-2009

Premessa: si tratta di una traccia di soluzione "buttata li" in poco tempo per consentire ai miei studenti di avere un termine di confronto con quanto hanno fatto stamattina durante la II prova.

punto 1)

si tratta di una realtà aziendale (una società telefonica) in cui un gruppo di **tecnici**, coordinati da un amministratore, gestiscono sul sito una non meglio precisata lista di "**contatti**". Questi potrebbero essere, ad esempio, dei consulenti che vengono chiamati all'occorrenza, oppure dei fornitori di attrezzature e servizi, o ancora dei clienti del servizio telefonico che hanno inoltrato richieste di intervento o infine persone che lavorano in aziende con le quali si interagisce per motivi vari.... In ogni caso il succo del problema è sempre lo stesso: accedere ad una rubrica di indirizzi condivisa on-line.

Trattandosi di dati sensibili occorre prevedere un meccanismo di autenticazione degli accessi, pertanto i tecnici che condividono la rubrica devono essere autenticati. Per far ciò loro stessi devono essere registrati in una anagrafica. Inoltre occorre poter sapere chi ha avuto accesso ai dati, quando e per quali ragioni lo ha fatto, quindi va prevista la possibilità di registrare gli **accessi** e le **operazioni** effettuate.

La traccia chiede anche di memorizzare la fine della sessione. Questo non è sempre realizzabile, in quanto se un utente non esce dal sito attraverso il link di "logout" (ad esempio perché chiude il browser) resta, dal punto di vista del sistema, ancora collegato, a meno che non si preveda un meccanismo di timeout sulle sessioni che disconnetta dopo un certo periodo di inattività. In tal caso non si avrebbe il dato della vera fine della sessione ma di quando il sistema ha scollegato l'utente. Comunque, ai fini di quanto chiesto dalla traccia, tutto ciò è ininfluente, si tratta solo di due attributi in una tabella

Infine, sempre in considerazione della delicatezza dei dati, le **variazioni** all'anagrafica devono essere confermate dall'amministratore del sistema.

Il dimensionamento del sistema non pone particolari problemi: sia i tecnici che i contatti potrebbero essere poche migliaia di righe al massimo, nell'ipotesi di una società a carattere nazionale. Il log degli eventi si presenta leggermente più impegnativo: nell'ipotesi di un migliaio di tecnici, che consultano la lista dei contatti 5 volte l'ora nell'arco delle otto ore di lavoro, potremmo avere 40 operazioni al giorno da registrare per ciascun tecnico, quindi, in un anno lavorativo 40 x 356 x 1000 = 14.600.000 operazioni registrate. Gli accessi al sistema, considerato che con un accesso ciascun tecnico può effettuare tutte le operazioni che vuole, potrebbero invece essere solo due o tre al giorno per ciascun tecnico, quindi possiamo stimare circa 500.000 accessi registrati all'anno. Anche con queste cifre, dato che per ciascuna operazione e ciascun accesso vanno registrate veramente poche informazioni (ad esempio, un centinaio di byte per operazione), possiamo pensare a tabelle di pochi gigabyte. Alla fine di ogni anno si può poi procedere ad archiviare i dati vecchi e ripartire con le tabelle degli

accessi e delle operazioni debitamente ripulite.

Per il resto, si tratta di un tradizionale server web che può essere preso in noleggio presso un provider qualsiasi. La tecnologia da utilizzare è quella tradizionale in questi casi: Apache, Php, MySQL.

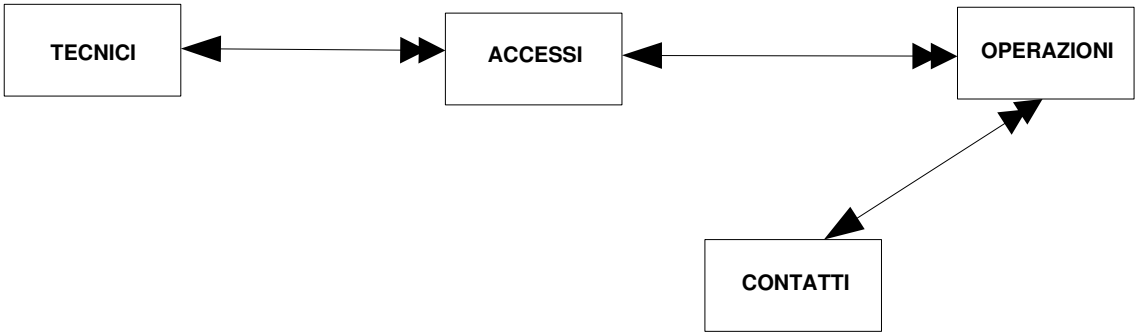
punto 2)

Quattro entità appaiono subito evidenti: ci sono TECNICI, che effettuano ACCESSI al sistema per effettuare OPERAZIONI che riguardano dei CONTATTI.

Considerato che:

- un tecnico può effettuare molti accessi al sistema, ma ciascun accesso è ovviamente riferito ad un solo tecnico
- con un solo accesso il tecnico può effettuare molte operazioni, ma ciascuna operazione è effettuata durante un accesso
- ogni operazione riguarda un singolo nominativo nella lista dei contatti, ma ciascun nominativo può essere consultato varie volte

si deduce immediatamente il seguente modello concettuale:

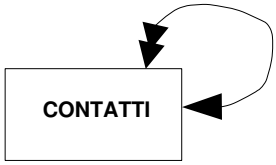


resta da risolvere il problema delle VARIAZIONI non confermate. Queste sono ovviamente riferite a nominativi presenti nella entità CONTATTI, e quindi devono essere poste in relazione con i contatti. Cosa sono le variazioni? Tutto sommato sono gli stessi attributi della tabella contatti, con un attributo ulteriore che implementi la integrità referenziale tra variazioni e contatti. Sicuramente abbiamo che:

- un contatto può ricevere diverse proposte di variazione
- ciascuna variazione è relativa ad un contatto

Detto così si possono scegliere due strade:

1. **associazione 1:N riflessiva.** Le variazioni sono nuove righe in CONTATTI che si riferiscono ad una riga già esistente.Un particolare attributo indica che non sono ancora confermate. Oppure, il fatto che si riferiscano ad un altra riga potrebbe già essere un indicatore di non avvenuta conferma, ma questo complicherebbe un tantino le query successive.



2. **associazione 1:N fra due entità differenti.** Questa è più semplice

da realizzare e gestire, ed inoltre si eviterebbe di inserire in CONTATTI attributi non necessari (chiave esterna e stato della variazione), e ciò è senz'altro meglio dal punto di vista del modello.

Va evidenziato che le variazioni di tipo "inserimento di un nuovo contatto", che non sarebbero riferite ad una riga già presente in CONTATTI, in entrambi i casi avrebbero un valore nullo nell'attributo di "chiave esterna".

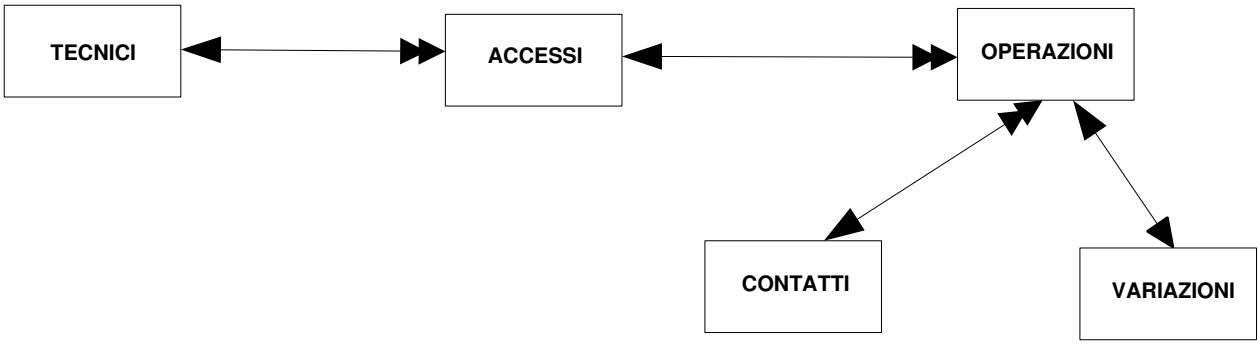
Veniamo alla conferma delle VARIAZIONI.

Nel primo caso (associazione riflessiva) si tratterebbe di eliminare la vecchia riga ed effettuare un update della nuova recuperando la chiave primaria della riga cancellata. Questo dovrebbe avvenire sempre in transazione.

Nel secondo caso occorrerebbe un update della riga da aggiornare recuperando gli attributi dalla riga di variazione.

Sin qui le due soluzioni sembrano equivalenti. Un elemento fa però propendere per la soluzione con due entità separate: in tale ipotesi risulta possibile collegare le VARIAZIONI alle OPERAZIONI, e ciò è importante, stando alla richiesta della traccia.

In conclusione, lo schema concettuale proposto è il seguente:



Lo schema logico, scegliendo di adottare il modello relazionale, è immediato. Non esistendo relazioni M:N non occorre inserire nuove tabelle, e quindi è sufficiente definire gli attributi e porre correttamente le chiavi esterne, tutte, ovviamente, nei rispettivi "lati N" delle associazioni. Si rimanda al punto 3) per vedere come viene realizzata la base dati

punto 3

Definizione delle relazioni in SQL. Si adotta una sintassi stile MySQL. Gli attributi, per comodità di esposizione, sono ridotti all'essenziale e in alcuni casi non sono neanche atomici (come l'atributo "indirizzo" in CONTATTI).

```
CREATE TABLE tecnici(
    id INT NOT NULL AUTO_INCREMENT,
    nome char(50),
```

```
    cognome char(50),
    username char(10) NOT NULL UNIQUE,
    password char(20) NOT NULL ,
    ruolo char,
    PRIMARY KEY (id)
)

CREATE TABLE contatti(
    id INT NOT NULL AUTO_INCREMENT,
    nome char(50),
    cognome char(50),
    telefono char(20) NOT NULL,
    indirizzo char(100) NOT NULL ,
    provincia char(2),
    denominazione char(50),
    annotazioni char(500),
    PRIMARY KEY (id)
)

CREATE TABLE accessi (
    id INT NOT NULL AUTO_INCREMENT,
    fk_tecnici INT NOT NULL,
    giorno_inizio DATE,
    orario_inizio TIME,
    giorno_fine DATE,
    orario_fine TIME,
    PRIMARY KEY (id),
    FOREIGN KEY (fk_tecnici) REFERENCES tecnici (id)
)

CREATE TABLE operazioni(
    id INT NOT NULL AUTO_INCREMENT,
    fk_accessi INT NOT NULL,
    fk_contatti INT NOT NULL,
    tipo char NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (fk_accessi) REFERENCES accessi (id),
    FOREIGN KEY (fk_contatti) REFERENCES contatti (id)
)

CREATE TABLE variazioni(
    id INT NOT NULL AUTO_INCREMENT,
    fk_operazioni INT NOT NULL,
    nome char(50),
    cognome char(50),
    telefono char(20) NOT NULL,
    indirizzo char(100) NOT NULL ,
    provincia char(2),
    denominazione char(50),
```

```
    annotazioni char(500),
    PRIMARY KEY (id),
    FOREIGN KEY (fk_operazioni) REFERENCES operazioni (id)
)
```

query 1)

```
    select * from contatti where provicia = 'PV' order by denominazione
```

query 2)

```
    select tecnici.nome, tecnici.cognome, data_inizio, ora_inizio
    from tecnici, accessi
    where tecnici.id = accessi.fk_tecnici
    and tecnici.username = 'paperino'
    order by data_inizio, ora_inizio
```

query 3)

Ci sono due modi:

1. utilizzo di tabella temporanea (due query, non necessitano di transazione)

```
        create temporary table giornalieri
        select count(*) acc_giorno
        from accessi
        where data_inizio between '01/01/2000' and '31/01/2000'
        group by data_inizio

        select avg(acc_giorno) from  giornalieri
```

2. solo con DBMS particolari (ORACLE, PostgreSql):

```
        select avg(acc_giorno)
        from (
            select count(*) acc_giorno
            from accessi
            where data_inizio between '01/01/2000' and '31/01/2000'
            group by data_inizio
        )
```

query 4)

Supponendo che l'attributo "tipo" nella tabella OPERAZIONI valga "N" per indicare un nuovo inserimento, si risolve così:

```
    select count(*), fk_tecnici from accessi, operazioni
    where accessi.id = operazioni.fk_accessi
    and operazioni.tipo = 'N'
    group by fk_tecnici
```

query 5)

Il fatto che il quesito dica "da ogni utente del gruppo di lavoro" fa pensare che non si tratti di un semplice elenco operazioni ma occorra anche evidenziare accanto all'operazione il nome del tecnico che l'ha eseguita. Occorre quindi un join fra tre tabelle.

```
select operazioni.*, tecnici.username
from tecnici, accessi, operazioni
where tecnici.id = accessi.fk_tecnici
and accessi.id = operazioni.fk_accessi
and data_inizio = '15/08/2008'
order by tecnici.username
```

query 6)
Molto simile alla 3). Creiamo una tabella temporanea che contiene il numero di consultazioni per ciascun contatto e poi da questa estraiamo il risultato. Attenzione, non è detto che il contatto più consultato sia uno solo, potrebbero esserci dei "pari merito"

```
create temporary table quanticontatti
select count(*) nume_operazioni, fk_contatti
from operazioni, accessi
where accessi.id = operazioni.fk_contatti
and data_inizio between '1/6/2009' and '7/6/2009'
group by fk_contatti

-----usiamo la temporanea per trovare i contatti più consultati

select contatti.*, nume_operazioni
from contatti, quanticontatti
where contatti.id = quanticontatti.fk_contatti and
nume_operazioni =(select max(nume_operazioni) from quanticontatti)
```

punto 4

Per il punto 4) procedere come durante le verifiche dell'anno (vedere al riguardo gli altri materiali sul sito):

1. definire le specifiche del sito web, con particolare riguardo alle funzionalità di amministrazione (login, gestione anagrafica delle tabelle, ottenimento di informazioni riassuntive, etc etc...)
2. carta di navigazione
3. carta di struttura dell'applicazione
4. codificare una parte in PHP (ad esempio la modifica della riga di una tabella oppure la visualizzazione in formato tabellare di tutte le righe di una tabella, generando i link alle singole righe per consentire di accedervi)