Esercizio di modellazione in SQL Server 2008: PARTE 1 creazione di un DB

Laboratorio Basi di Dati

Laura Po

Specifiche (tratto dal testo dell'appello BDATI 02/07/2009)

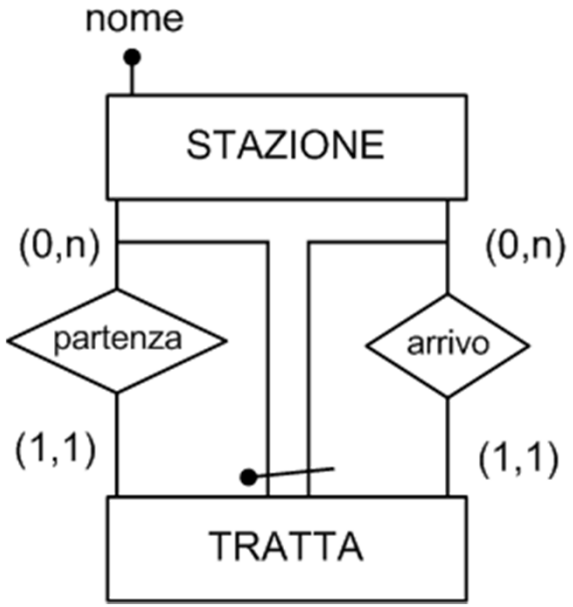
- Si vuole progettare un database per la gestione delle ferrovie dello stato.
- I treni gestiti sono identificati da un numero. Su ciascun treno sono specificate le classi per le quali offre servizio (prima, seconda, lusso).
- Le tratte che un treno può percorrere sono identificate dalla stazione di partenza e dalla stazione di arrivo. Ogni stazione è identificata da un nome. Ogni treno percorre diverse tratte in date e ore distinte. Il sistema memorizza le tratte percorse dai treni.
- Gli utenti sono identificati attraverso un username, hanno una password, una email ed eventualmente una tessera fedeltà. Un utente può acquistare biglietti per una tratta per uno specifico numero di posti ad un prezzo stabilito dal sistema.
- Il sistema informativo permette di definire particolari utenti di classe business che usufruiscono di una percentuale di sconto nell'acquisto di biglietti.

Modellazione ER

- Si vuole progettare un database per la gestione delle ferrovie dello stato.
- I treni gestiti sono identificati da un numero. Su ciascun treno sono specificate le classi per le quali offre servizio (prima, seconda, lusso).

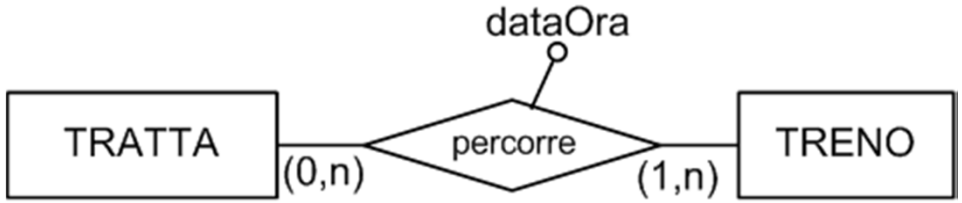
Modellazione ER

- Le tratte che un treno può percorrere sono identificate dalla stazione di partenza e dalla stazione di arrivo. Ogni stazione è identificata da un nome.

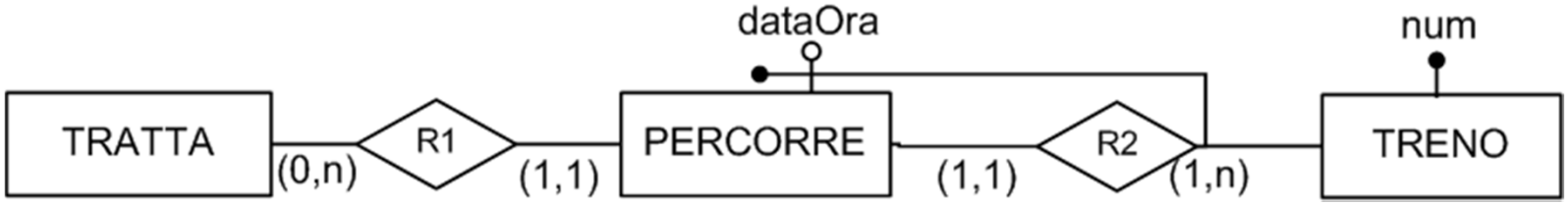


Modellazione ER

- Ogni treno percorre diverse tratte in date e ore distinte. Il sistema memorizza le tratte percorse dai treni.

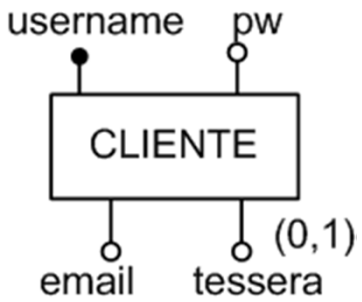


- Occorre *reificare* per imporre che nella stessa data e ora il treno non percorra più di una tratta



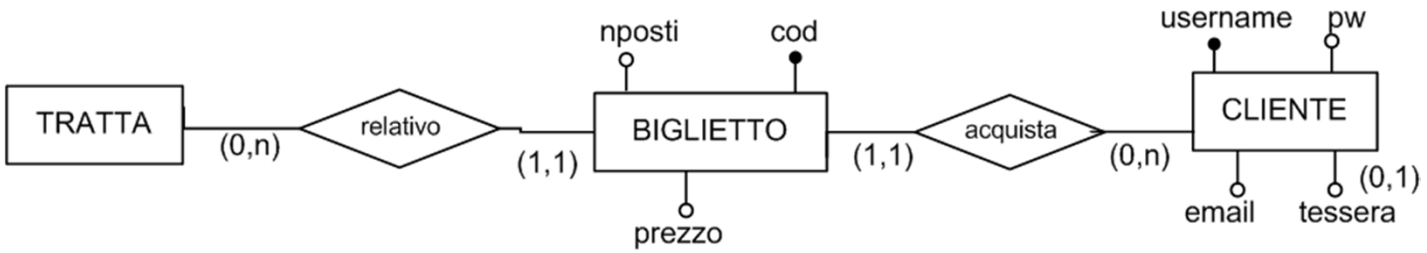
Modellazione ER

- Gli utenti sono identificati attraverso un username, hanno una password, una email ed eventualmente una tessera fedeltà.



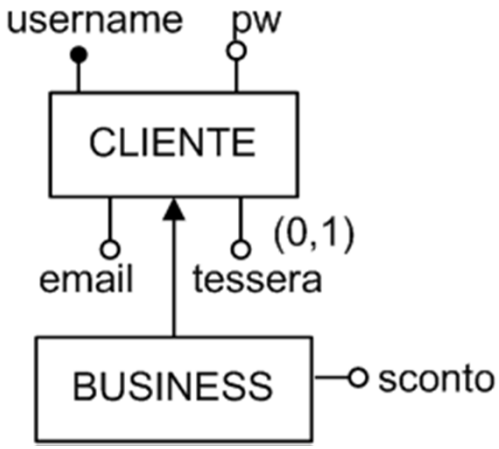
Modellazione ER

- Un utente può acquistare biglietti per una tratta per uno specifico numero di posti ad un prezzo stabilito dal sistema.

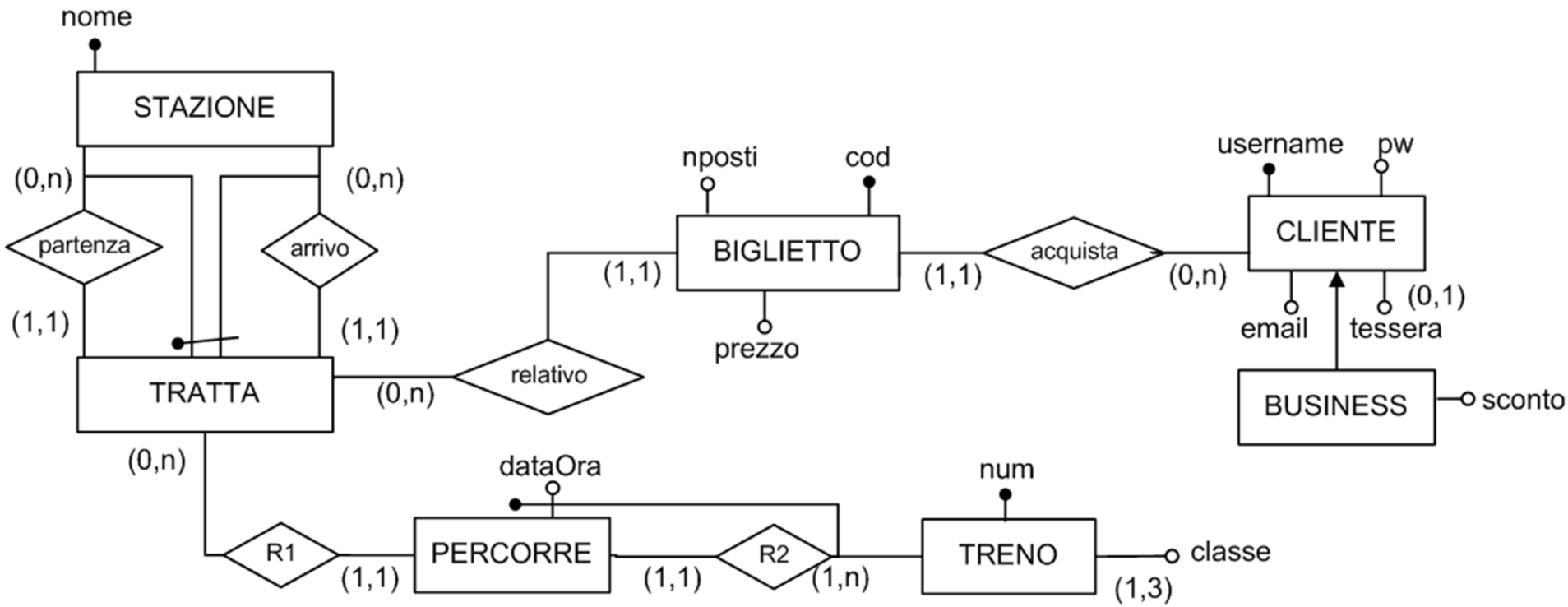


Modellazione ER

- Il sistema informativo permette di definire particolari utenti di classe business che usufruiscono di una percentuale di sconto nell'acquisto di biglietti.

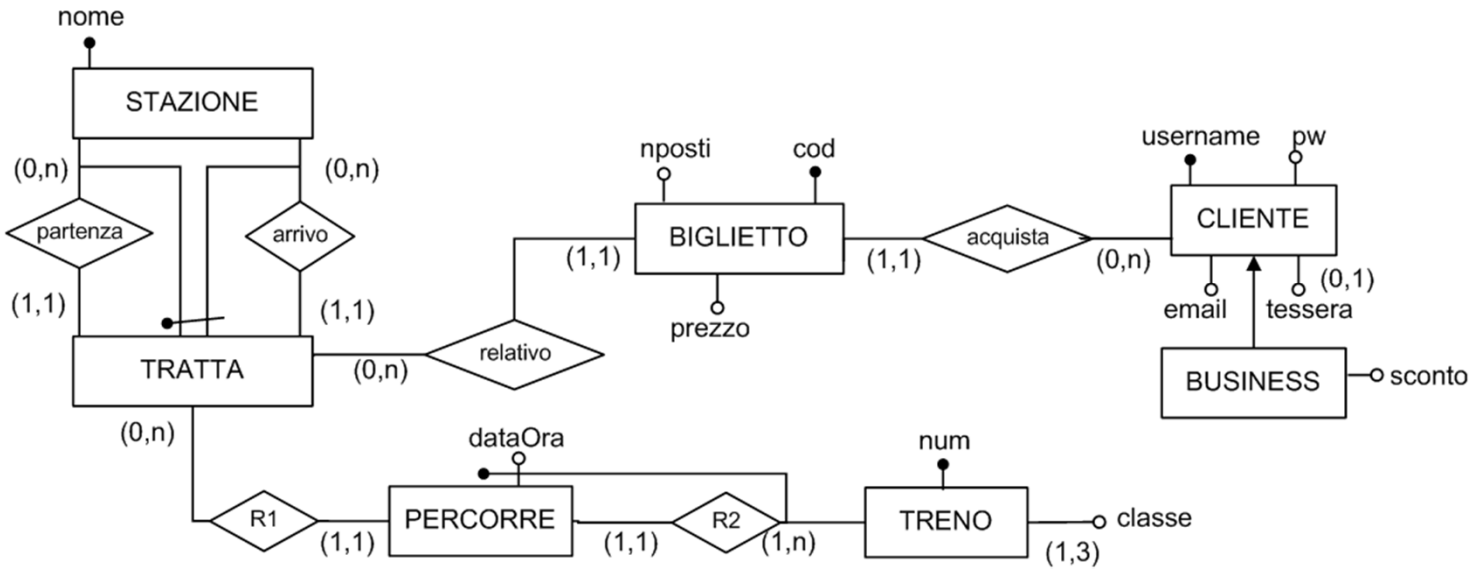


ER finale

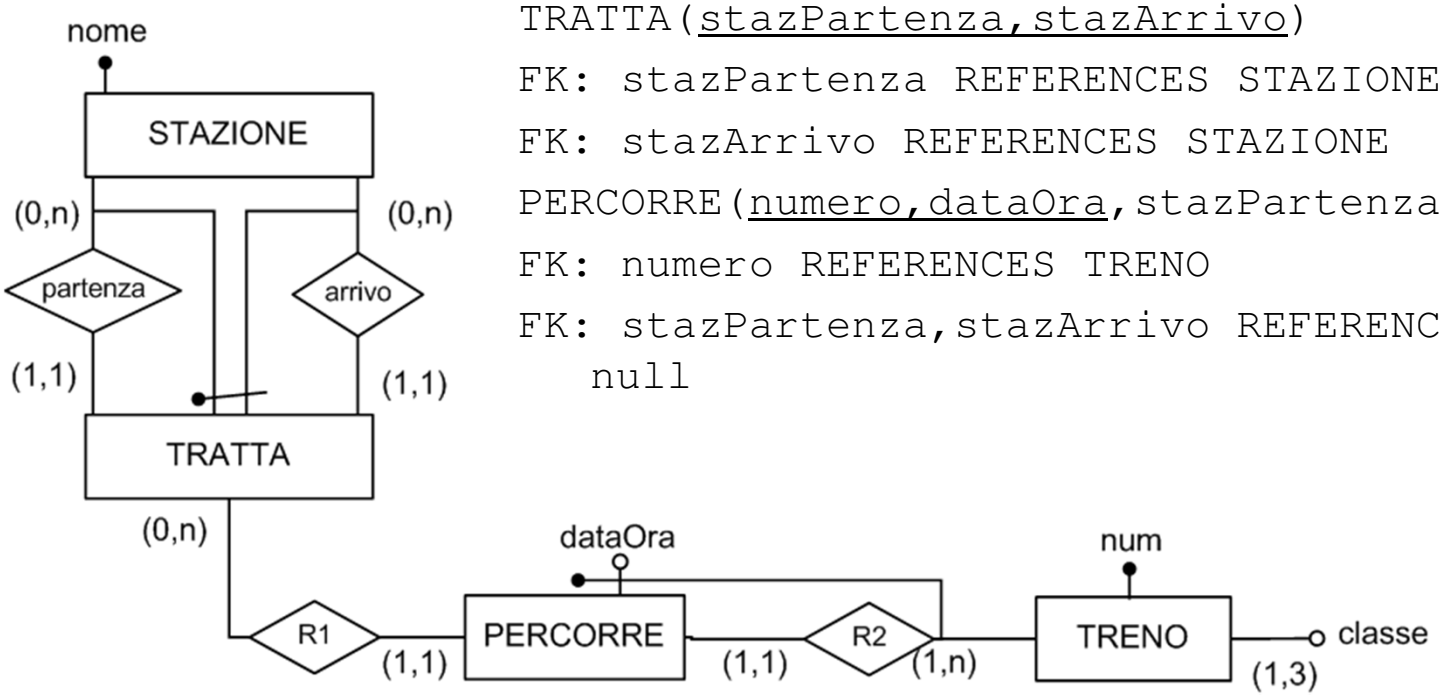

nome
STAZIONE
(0,n)
(0,n)
partenza
arrivo
(1,1)
(1,1)
TRATTA
(0,n)
relativo
nposti
cod
BIGLIETTO
(1,1)
(1,1)
prezzo
acquista
(0,n)
username
pw
CLIENTE
email
tessera
(0,1)
BUSINESS
sconto
(0,n)
dataOra
num
R1
(1,1)
PERCORRE
(1,1)
R2
(1,n)
TRENO
classe
(1,3)

Modello logico

- A partire dallo schema ER modellare lo schema relazionale
- (10 min)



Progetto logico relazionale



TRENO(<u>numero</u>,classe1,classe2,classe3)

STAZIONE(<u>nome</u>)

TRATTA(<u>stazPartenza,stazArrivo</u>)

FK: stazPartenza REFERENCES STAZIONE

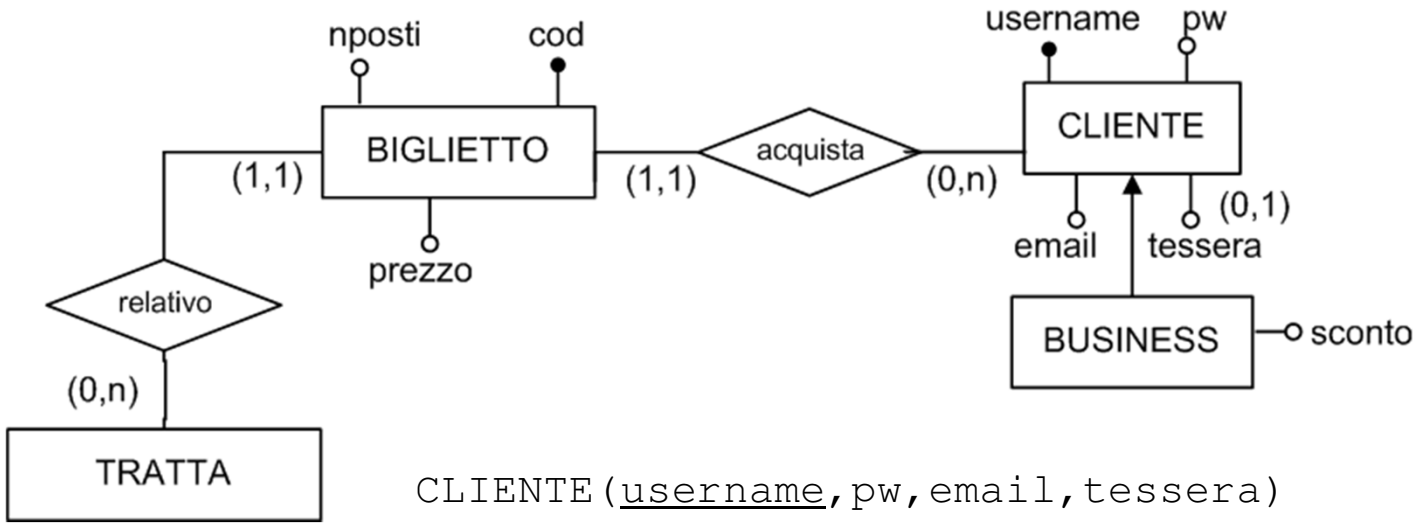
FK: stazArrivo REFERENCES STAZIONE

PERCORRE(<u>numero,dataOra</u>,stazPartenza,stazArrivo)

FK: numero REFERENCES TRENO

FK: stazPartenza,stazArrivo REFERENCES TRATTA not null

Progetto logico relazionale



CLIENTE(<u>username</u>,pw,email,tessera)

BUSINESS(<u>username</u>,sconto)

FK: username REFERENCES CLIENTE

BIGLIETTO(<u>cod</u>,nposti,prezzo,username,
stazPartenza,stazArrivo)

FK: stazPartenza,stazArrivo REFERENCES TRATTA
not null

FK: username REFERENCES CLIENTE not null

SCRIPT di creazione del DB

```
TRENO(numero,classe1,classe2,classeL)
STAZIONE(nome)
TRATTA(numTratta,stazPartenza,stazArrivo)
AK: stazPartenza,stazArrivo
FK: stazPartenza REFERENCES STAZIONE
FK: stazArrivo REFERENCES STAZIONE
PERCORRE(numero,dataOra,numTratta)
FK: numero REFERENCES TRENO
FK: numTratta REFERENCES TRATTA
CLIENTE(username,pw,email,tessera)
BUSINESS(username,sconto)
FK: username REFERENCES CLIENTE
BIGLIETTO(cod,nposti,prezzo,username,numTratta)
FK: numTratta REFERENCES TRATTA
FK: username REFERENCES CLIENTE
```

SET DATEFORMAT (Transact-SQL)

- Imposta l'ordine delle parti della data relative a mese, giorno e anno per l'interpretazione di stringhe di caratteri **date**, **smalldatetime**, **datetime**, **datetime2** e **datetimeoffset**.

Sintassi

SET DATEFORMAT { format | @format_var }

Argomenti

format | *@format_var*

Ordine delle parti della data. I parametri validi sono **mdy**, **dmy**, **ymd**, **ydm**, **myd** e **dym**. L'impostazione predefinita per l'inglese (Stati Uniti) è **mdy**.

Osservazioni

- DATEFORMAT **ydm** non è supportata per i tipi di dati **date**, **datetime2** e **datetimeoffset**.

SET DATEFORMAT (Transact-SQL)

- L'opzione SET DATEFORMAT viene impostata in fase di esecuzione, non in fase di analisi.
- L'opzione SET DATEFORMAT ignora l'impostazione esplicita del formato di data dell'opzione SET LANGUAGE.
- L'effetto dell'impostazione DATEFORMAT sull'interpretazione di stringhe di caratteri potrebbe essere diverso per i valori **datetime** e **smalldatetime** rispetto ai valori **date**, **datetime2** e **datetimeoffset**, a seconda del formato di stringa. Questa impostazione influisce sull'interpretazione di stringhe di caratteri nel momento in cui queste vengono convertite in valori di data per l'archiviazione nel database. Non influisce sulla visualizzazione di valori del tipo di dati date archiviati nel database o sul formato di archiviazione di questi.

SCRIPT CREAZIONE DB

- Treno – impostiamo gli attributi classe come 1°, 2° e lusso, dobbiamo controllare che un treno definisca almeno una classe.

```
-- Treno

CREATE TABLE Treno
        (
        numero varchar(10) PRIMARY KEY,
        classe1 BIT not null,
        classe2 BIT not null,
        classeL BIT not null,
        CHECK (classe1=1 or classe2=1 or classeL=1)
        )
GO
```

SCRIPT CREAZIONE DB

- Tratta – dobbiamo definire i controlli opportuni per la AK

```
-- Tratta

CREATE TABLE Tratta
        (
        numtratta varchar(5) PRIMARY KEY,
        stazpartenza VARCHAR(20) REFERENCES Stazione not null,
        stazarrivo VARCHAR(20) REFERENCES Stazione not null,
        UNIQUE (stazpartenza,stazarrivo)
        )
GO
```

SCRIPT CREAZIONE DB

- Percorre

```
-- Percorre

CREATE TABLE Percorre
        (
        numero varchar(10) REFERENCES Treno not null,
        data DATETIME not null,
        numtratta varchar(5) REFERENCES Tratta not
null,
        PRIMARY KEY (numero,data)
        )
GO
```

SCRIPT CREAZIONE DB

- Cliente e Business

```
-- Cliente

CREATE TABLE Cliente
        (
        username varchar(20) PRIMARY KEY,
        pw varchar(10) not null,
        email varchar(40) not null,
        tessera varchar(10) null
        )
GO

-- Business

CREATE TABLE Business
        (
        username varchar(20) PRIMARY KEY REFERENCES Cliente,
        sconto int not null
        )
GO
```

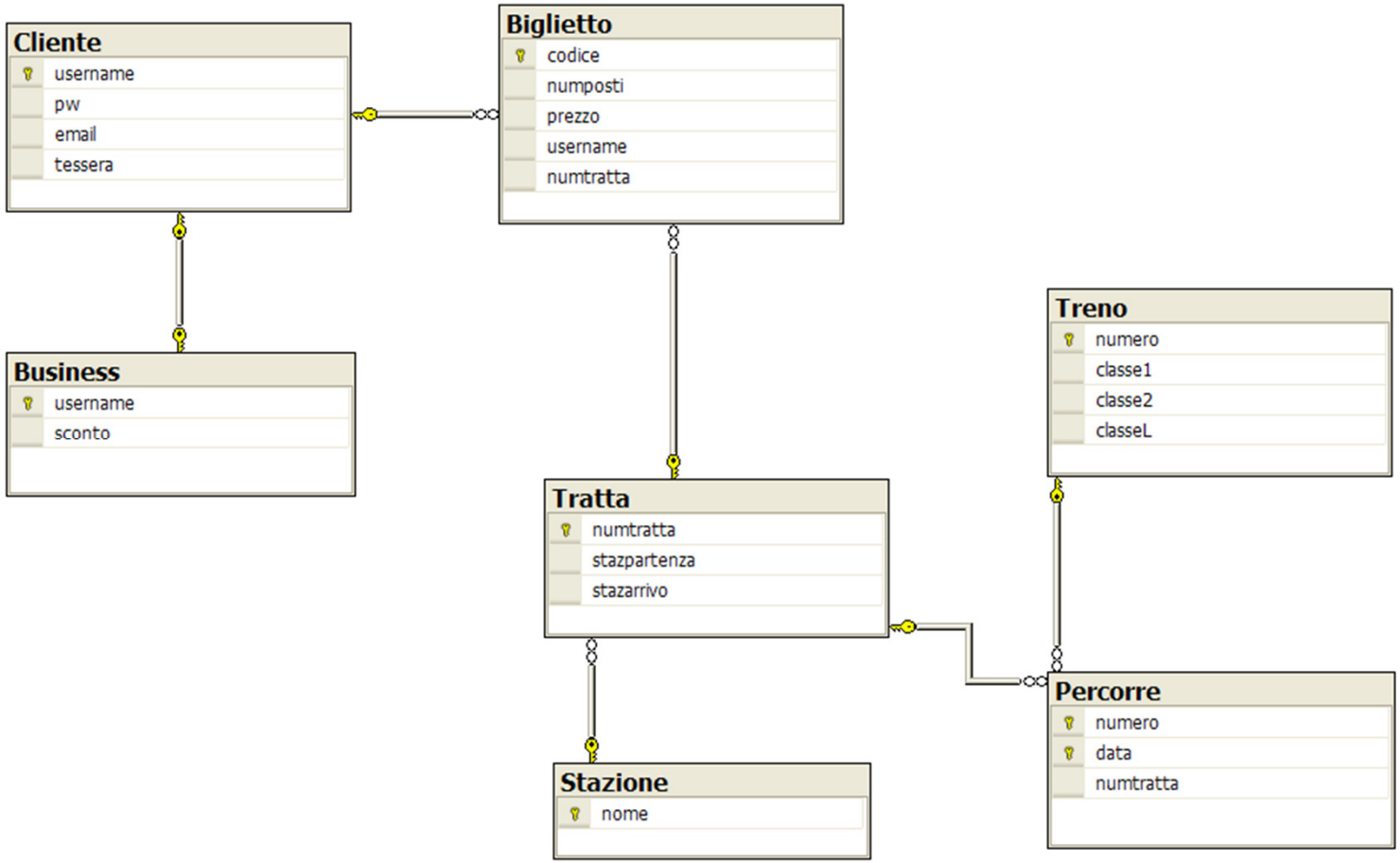
SCRIPT CREAZIONE DB

- Biglietto

```
-- Biglietto

CREATE TABLE Biglietto
        (
        codice INT IDENTITY PRIMARY KEY,
        numposti INT not null,
        prezzo money not null,
        username varchar(20) REFERENCES Cliente not null,
        numtratta varchar(5) REFERENCES Tratta not null
        )
GO
```

Struttura DB



Analisi del funzionamento delle FOREIGN KEY

- Se si esegue la seguente operazione sul database popolato

```
UPDATE Tratta
set numtratta='MO:BO'
where numtratta='MO-BO'
```

- Si ottiene un messaggio di errore simile al seguente

```
Msg 547, Level 16, State 0, Line 3
The UPDATE statement conflicted with the REFERENCE constraint
  "FK__Percorre__numtra__1367E606". The conflict occurred in database
  "Test21Marzo2011", table "dbo.Percorre", column 'numtratta'.
The statement has been terminated.
```

- Questo avviene perché il sistema ha rilevato la *VIOLAZIONE DEL VINCOLO DI INTEGRITA'*, ovvero esistono delle istanze in tabelle dipendenti (in questo caso nelle tabelle `PERCORRE` e `BIGLIETTO`) che fanno riferimento all'istanza di tabella `TRATTA` che stiamo cercando di modificare.

Modifica del funzionamento delle FOREIGN KEY

- E' possibile cambiare il funzionamento della foreign key in seguito al verificarsi di una violazione al vincolo di integrità referenziale.
- La clausola ON UPDATE, utilizzata in fase di definizione del vincolo di FOREIGN KEY, definisce che azione intraprendere quando si cerca di aggiornare una chiave candidata verso la quale esistono riferimenti di foreign key.
- Esempio, modificare gli script nel seguente modo

```
CREATE TABLE Percorre
    (
    numero varchar(10) REFERENCES Treno not null,
    data DATETIME not null,
    numtratta varchar(5) REFERENCES Tratta ON UPDATE CASCADE,
    PRIMARY KEY (numero,data)
    )
CREATE TABLE Biglietto
    (
    codice INT IDENTITY PRIMARY KEY,
    numposti INT not null,
    prezzo money not null,
    username varchar(20) REFERENCES Cliente not null,
    numtratta varchar(5) REFERENCES Tratta ON UPDATE set null
    )
```

Modifica del funzionamento delle FOREIGN KEY

- Se ora ritorno ad eseguire l'operazione di modifica sul database popolato

```
UPDATE Tratta
set numtratta='MO:BO'
where numtratta='MO-BO‘
```

- non riscontro più un errore!

- L'aggiornamento verrà eseguito sulla tabella `Tratta` e sarà propagato anche sulle istanze della tabella `Percorre`, che fanno riferimento all'istanza modificata sulla tabella `Tratta`. In tabella `Biglietto`, invece, in corrispondenza di istanze che fanno riferimento a quella modificata verrà posto `numtratta` a `null`

Esercizio di modellazione in SQL Server 2008: PARTE 2 interrogazione e modifica

Laboratorio Basi di Dati

Laura Po

Interrogazioni

1. Selezionare il cliente che ha acquistato più biglietti
2. Selezionare il treno che ha percorso più tratte dal 1/01/2009 al 05/01/2009
3. Selezionare la tratta più percorsa e visualizzare il numero di treni medio che la percorre giornalmente
4. Selezionare il cliente di classe BUSINESS che ha acquistato più biglietti

Interrogazioni

5. Visualizzare per ciascun cliente i suoi dati e tutti i biglietti acquistati.

6. Visualizzare per ciascun cliente i suoi dati e tutti i biglietti acquistati, compresi i clienti che non hanno acquistato alcun biglietto.

7. Selezionare per ciascun cliente e per ciascun biglietto da lui acquistato i possibili treni che percorrono la tratta a cui si riferisce il biglietto.

Interrogazioni

8. Selezionare per ciascuna tratta il cliente che ha comprato il biglietto al minor prezzo, comprese le tratte per cui non sono stati acquistati biglietti.

9. Selezionare clienti che hanno acquistato biglietti per una tratta per cui esista almeno un altro biglietto acquistato da un'altro cliente.

10. Visualizzare per ciascun cliente il prezzo medio dei biglietti acquistati su ciascuna tratta.

Inserimenti/modifiche

a) Per ogni cliente di classe business che possiede uno sconto inferiore al 5%, impostare lo sconto pari al 5%

b) *Offerta promozionale* – Inserire un biglietto GRATUITO per due posti sulla tratta Modena-Bologna per ciascun cliente che possiede una tessera

Modifiche al DB

c) Aggiungere l'informazione relativamente alla data di acquisto dei biglietti
d) Aggiungere un prezzo minimo e massimo sulle tratte calcolato come il prezzo minimo e massimo pagato per l'acquisto di biglietti sulle relative tratte
e) Aggiungere il vincolo che la stazione di partenza sia diversa dalla Stazione di arrivo sulla tabella TRATTA