BASI DI DATI
-PROGETTO-

di Bizzaro Francesco

ABSTRACT

“Bagni Piovra” è un nuovo stabilimento balneare del lido di Sottomarina. Essendo al passo con i tempi, offre la possibilità di prenotare uno o più ombrelloni - con tutti gli accessori annessi (lettini, sedie) - sia sul posto, sia da casa attraverso il sito web. Inoltre lo stabilimento mette a disposizione delle cabine per depositare oggetti e dei pedalò per uso dei clienti.
Sono presenti 156 ombrelloni, suddivisi in 6 file da 26, ed in tre fasce in base alla vicinanza al mare: la prima fascia comprende le 2 file più vicine all'acqua, la seconda le successive due ed infine la terza comprende le 2 file rimanenti.
Si vuole realizzare un sistema informatico per la gestione delle prenotazioni delle varie risorse offerte. In particolare si vuole dare la possibilità ai clienti di prenotare un ombrellone sia dalla struttura in sede, sia direttamente dal sito online. Durante la prenotazione il cliente potrà scegliere il singolo ombrellone che preferisce. Per gli amministratori deve esserci la possibilità di definire le varie tariffe da applicare alla prenotazione, a seconda del periodo di stagione, della durata, della posizione dell'ombrellone ed eventualmente dei lettini scelti.
Le cabine, per chi lo desidera, vengono prenotate assieme agli ombrelloni.
I pedalò invece vengono noleggiati solo sul momento.
Il database deve gestire le prenotazioni in modo da evitare sovrapposizioni di date. Si devono rendere possibili anche prenotazioni multiple di più ombrelloni.

DESCRIZIONE DEI REQUISITI

Si vuole realizzare una base di dati per la gestione di uno stabilimento balneare che vuole offrire ai clienti la possibilità di prenotazione online, in aggiunta alla normale prenotazione in sede.
Si vuole tenere traccia degli ombrelloni presenti; in particolare ciascuno di essi appartiene ad una fila, ed è identificato da un numero all'interno di essa.
Le file sono raggruppare a loro volta in 3 fasce distinte, caratterizzate dalla diversa vicinanza al mare. Ad ogni fascia è associato un prezzo giornaliero specifico, che dipende anche dal mese e dall'anno di riferimento. Questo prezzo può essere modificato nel tempo dall'amministratore dello stabilimento. Lo stabilimento dispone di 156 ombrelloni, divisi in 6 file da 26.
Oltre all'ombrellone vi sono altri accessori di cui si vuole tenere conto, ovvero lettini, sedie e cabine. Di questi accessori si vuole conoscere il prezzo di noleggio giornaliero, il numero di essi che non è in uso in un dato momento (per non assegnarne più di quanti se ne dispone) e quali di essi siano invece associati ad un cliente.
In totale si dispone di 70 cabine, 150 sedie e 350 lettini.
Gli utenti del database si distinguono tra dipendenti dello stabilimento e clienti. I dipendenti necessitano di una coppia username-password per potersi autenticare e accedere ai servizi. I clienti si differenziano a loro volta tra clienti “normali” e clienti “online”. In entrambi i casi si vogliono memorizzare il nome e cognome, mentre per i clienti online è necessario anche un indirizzo mail, uno username ed una password per il login (per scelta organizzativa non si vuole richiedere il codice fiscale ad ogni cliente che faccia una prenotazione).
Una prenotazione (sia normale che online) deve tenere traccia del cliente che l'ha effettuata, delle date di inizio e fine, dell'ombrellone (o degli ombrelloni) che è stato assegnato, e degli eventuali accessori che vi si possono associare. Si deve calcolare automaticamente il prezzo totale da mettere in fattura, in funzione dei dati precedenti. In particolare ogni ombrellone ha un prezzo per giornata (in base alla localizzazione) che può variare a seconda del giorno, e ogni accessorio ha un suo prezzo fisso (sempre per giornata). Si possono effettuare prenotazioni multiple.
Gli accessori si possono associare agli ombrelloni per dei periodi di tempo entro i limiti della prenotazione dell'ombrellone, ma che possono anche non coincidere (per esempio si può volere un lettino in più solo per un giorno..). Degli accessori possono essere aggiunti in qualsiasi momento.
Un cliente “online” può fare una prenotazione solo se questa copre un periodo di tempo entro l'anno corrente.

Durante la giornata i gestori dello stabilimento devono poter leggere le informazioni sullo stato attuale di occupazione degli ombrelloni (ad esempio la mattina per distribuire i lettini..). Queste operazioni si verificano in media 5 volte al giorno.
Per avere un resoconto dell'andamento dei guadagni (mediamente consultato 1 volta alla settimana), si vuole tener traccia dei pagamenti effettuati dai vari clienti. In particolare un pagamento rappresenta semplicemente una transazione di denaro, ed è caratterizzato dal cliente che lo compie, una data, un importo e una causale che ne specifica il motivo.

Si vuole memorizzare anche l'utilizzo dei pedalò messi a disposizione dalla struttura. Questi sono caratterizzati dal numero di persone che possono contenere, ed hanno un loro numero di matricola univoco. Si possono noleggiare per un numero variabile di ore, ma solo nel momento dell'utilizzo (non si prenotano da internet). È associato un costo orario fisso ad ogni pedalò. Solo chi è cliente dello stabilimento può accedere a questo servizio. Al momento lo stabilimento dispone di 2 pedalò da quattro posti e 3 da due posti.

PROGETTAZIONE CONCETTUALE

Glossario dei termini principali

Termine	***Descrizione***	***Collegamenti***
Ombrellone	Ombrelloni messi a disposizione, identificati da numero e riga. Appartenenti ad una fascia, la quale ne determina il prezzo giornaliero, a seconda del periodo.	Fascia, Prenotazione
Utente	Può essere un dipendente dello stabilimento che consulta la base di dati, o un cliente che effettua una prenotazione.	Cliente
Cliente	Colui che prenota almeno un ombrellone. Può essere online.	Prenotazione, Pagamento
Accessorio	Sono degli optional che possono essere associati ad ogni ombrellone per un intervallo temporale. Possono essere cabine, lettini o sedie.	Prenotazione
Pagamento	Tiene conto delle transazioni monetarie: data di esecuzione, cliente, importo e causale.	Cliente
Prenotazione	Quando un cliente prenota uno o più ombrelloni in un dato periodo di tempo (eventualmente associa degli accessori).	Cliente, Ombrellone, Accessorio

Elenco delle principali operazioni previste

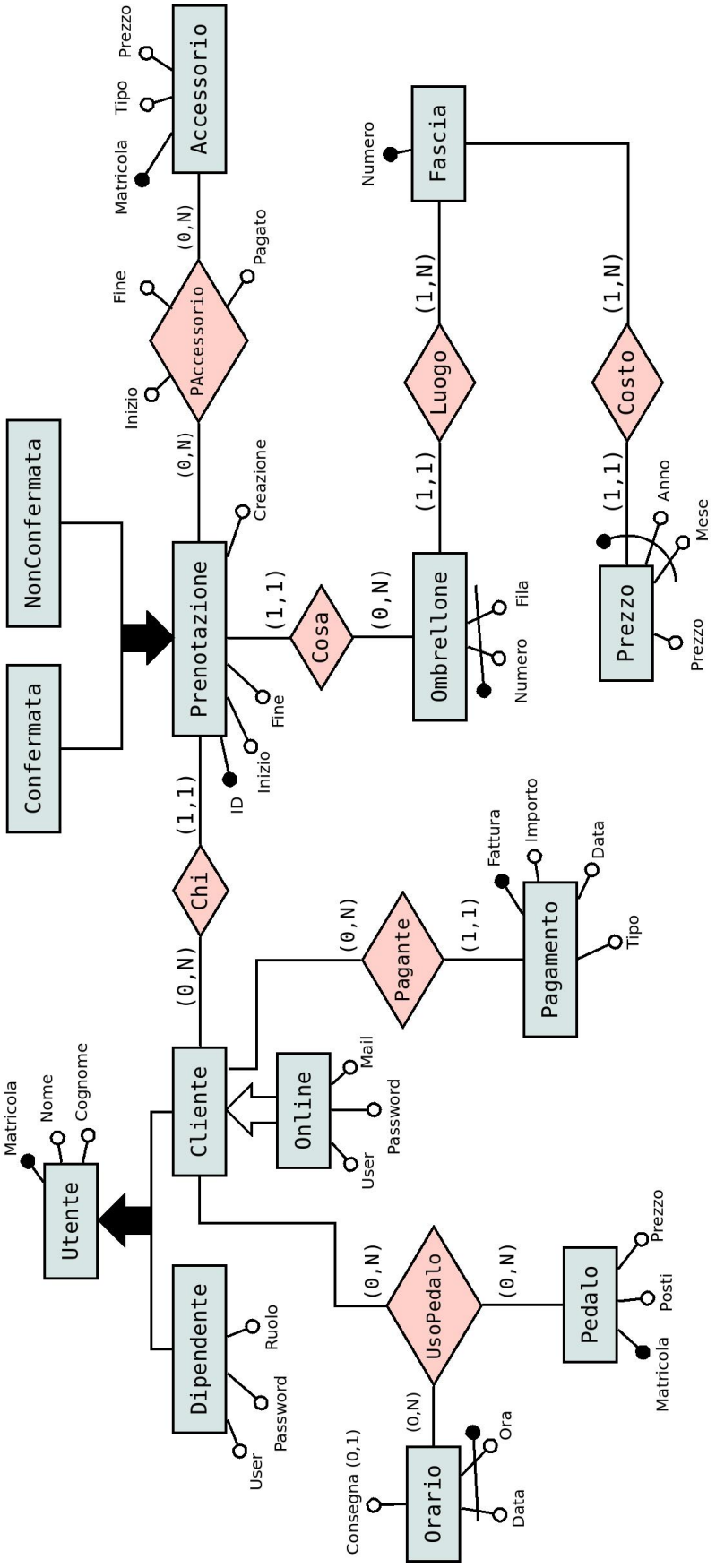
#	***Frequenza***	***Operazione***
Op.1	2 v. all'anno	Aggiornamento/modifica tariffe ombrelloni
Op.2	11 v. al giorno	Prenotazione ombrellone e accessori connessi (dopo verifica disponibilità)
Op.3	30 v. al giorno	Emissione fattura (con calcolo costo totale)
Op.4	15 v. al giorno	Prenotazione pedalò
Op.5	5 v. al giorno	Lettura occupazione ombrelloni
Op.6	1 v. alla settimana	Lettura andamento guadagni
Op.7	1 v. al giorno	Lettura occupazione/disponibilità accessori

Creazione dello schema E-R

In base alle specifiche, è stato modellato uno schema Entity-Relationship, utilizzando l'approccio inside-out.
Una prima entità rappresenta il concetto di “utente”. Essa possiede gli attributi Nome e Cognome, ed è specificata da una generalizzazione totale in “Dipendente” e “Cliente”. Per i dipendenti vengono aggiunti gli attributi “User”, “Password” e “Ruolo”, mentre per i Clienti bisogna distinguere tra quelli “online” e quelli normali, pertanto viene creato un sottoinsieme dell'entità. Quest'ultimo avrà dei dati aggiuntivi: un indirizzo mail, e una coppia user-password per poter fare la login nel sito.
Dato che un cliente può effettuare delle prenotazioni, si passa ora a specificare queste: sono composte da un unico cliente, un unico ombrellone[1], una data di inizio e una data di fine prenotazione. L'entità Prenotazione è dunque legata all'entità Cliente con una relazione uno a molti, e analogamente con l'entità Ombrellone di cui si specificheranno le caratteristiche in seguito. Per gestire meglio il pagamento online della prenotazione, è stato deciso di introdurre una generalizzazione in “Confermata” e “NonConfermata”: solo dopo il pagamento effettivo, una prenotazione è confermata, mentre una prenotazione non confermata rimane nella base di dati solamente per un periodo limitato, per consentire la transizione del denaro senza rischio di race conditions. Viene aggiunto inoltre l'attributo “Creazione” che specifica il momento preciso in cui la prenotazione è stata fatta.
Gli Ombrelloni sono caratterizzati da una fila e un numero identificativo al suo interno. Sono in relazione uno a molti con le Fasce, le quali sono state separate in entità a sé poiché contribuiscono ad identificare i Prezzi degli ombrelloni. L'entità Prezzo è composta dagli attributi anno, mese e importo. La chiave è l'insieme di mese, anno e fascia con il costrutto di identificazione esterna.
Tornando alle prenotazioni, ad esse possono essere associati degli Accessori. Per praticità nel rappresentare gli accessori, è stato scelto di considerarli singolarmente come enti[2], anziché limitarsi ad indicarne tipo e quantitativo disponibile. In questo modo si possono raccogliere in una unica entità sia le cabine che i lettini e le sedie, ma sopratutto possono essere facilmente gestite le prenotazioni per intervalli di tempo che non coincidono con quelli dell'ombrellone. Un accessorio inteso in questa maniera è caratterizzato da una matricola[3], un prezzo e il tipo di oggetto. Gli accessori sono nella relazione molti a molti “PAccessorio” con le prenotazioni, per permettere prenotazioni multiple; in aggiunta possono essere specificate delle date di inizio e fine, purché all'interno del range della prenotazione di riferimento.
I Pagamenti (transazioni) sono resi attraverso una entità contenente gli attributi: fattura (che rappresenta in realtà solo una chiave e non ha relazioni con le fatture emesse dallo stabilimento), data, importo e tipo. Questa entità è in relazione uno a molti con i Clienti.
Per concludere restano da gestire i pedalò, la cui entità prevede una matricola, il numero di posti e il prezzo. Per gestire il loro utilizzo è stata introdotta una relazione molti a molti ternaria fra le entità Cliente, Pedalo e Orario (la quale rappresenta un intervallo di tempo).
Sono stati introdotti degli attributi identificatori aggiuntivi in alcune entità, per semplificarne l'identificazione: matricola per Utente, e ID per Prenotazione.
Per semplificare la verifica della riconsegna dei pedalò, e del pagamento degli accessori sono stati infine aggiunti gli attributi “consegna” per Orario e “pagato” per Paccessorio.

1 Ogni ombrellone si prenota a sé, ma nulla vieta che uno stesso cliente prenoti più ombrelloni.
2 In modo analogo agli ombrelloni.
3 Che solo nelle cabine corrisponde ad un ente fisico specifico.

Dall'analisi precedente, si ricava il seguente schema Entity-Relationship:



PROGETTAZIONE LOGICA

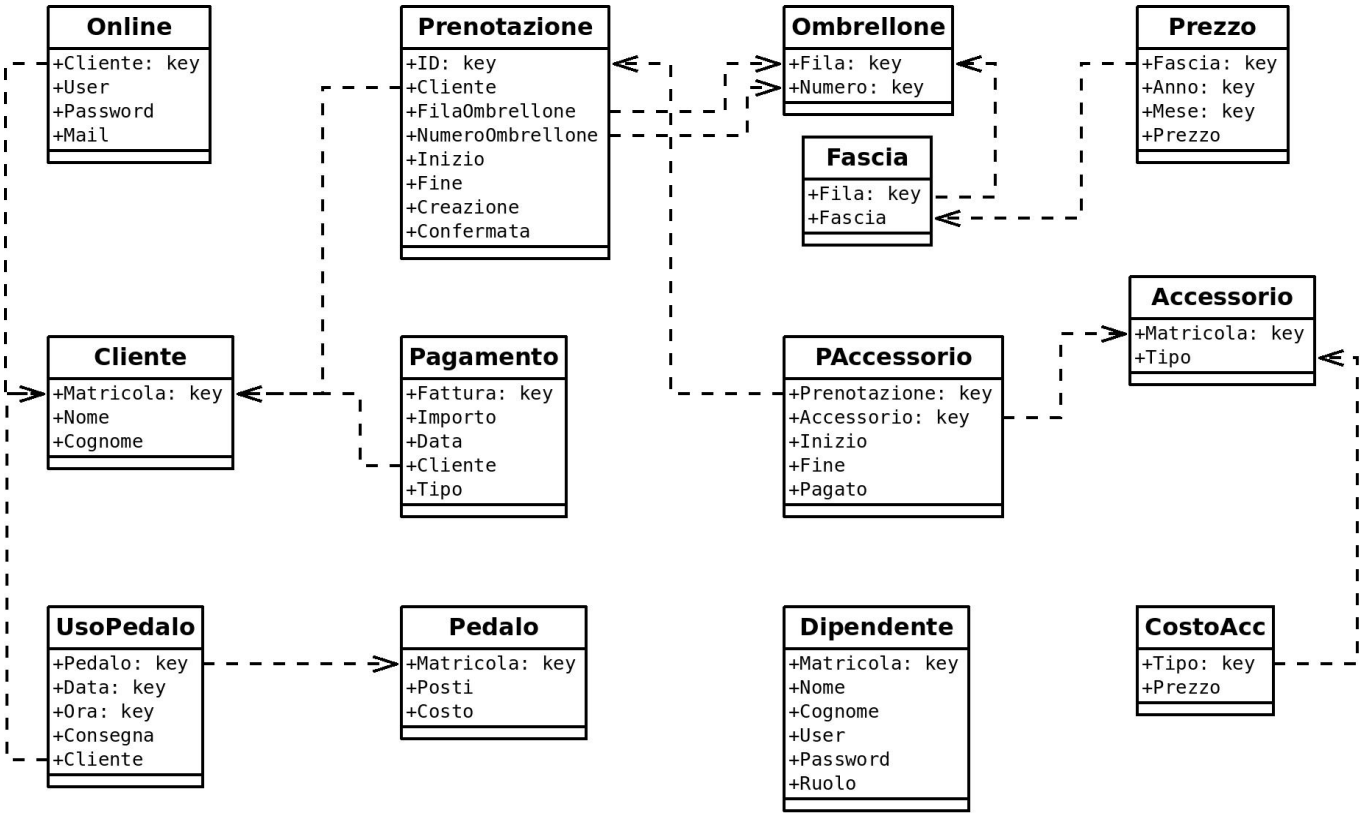
Tavola dei volumi[4]

Concetto	*Volume*	*Tipo*
Cliente	2000	E
Ombrellone	156	E
Accessorio	570	E
Fascia	3	E
Prezzo	30	E
Pedalo	5	E
Pagamento	10000	E
Prenotazione	5000	R
PAccessorio	5000	R
UsoPedalo	3000	R

Ristrutturazione dello schema E-R e successiva traduzione
Nello schema Entity-Relationship precedente non sono presenti ridondanze causate da attributi. L'unica possibile ridondanza consiste nell'identificazione delle fasce cui gli ombrelloni possono appartenere, le quali potrebbero essere ricavate direttamente dalla fila dell'ombrellone. Tuttavia è stato scelto di separare l'entità Fascia per poter associare più facilmente i prezzi agli ombrelloni, i quali sono riferiti alle fasce appunto, e non alle file. Mantenere l'entità Fascia non costa molto dato che vi saranno solamente 3 fasce, al contrario scrivere il prezzo in funzione delle file implicherebbe un uso maggiore di memoria, poiché le file sono più delle fasce (il doppio). Si noti inoltre che se nella traduzione le fasce vengono accorpate nello schema di relazione Ombrellone, il numero di accessi per trovare la fascia e quello per trovare la fila di uno stesso ombrellone non cambia.
Si passa ora all'eliminazione delle generalizzazioni. Per eliminare la generalizzazione di Utente è stato scelto di accorpare il genitore nelle entità figlie. Questo essenzialmente per due motivi: tenere separati i dipendenti dai clienti della struttura, e ridurre il numero di accessi nelle interrogazioni che riguardano i clienti. Solo i clienti infatti partecipano nelle relazioni Pagante, Chi e UsoPedalo. Per gestire i dati aggiuntivi dei clienti Online è stato scelto, ancora una volta, di tenere le due entità separate. Unirle causerebbe infatti l'introduzione in Cliente di 3 attributi con possibile valore null. D'altro canto mantenendo la distinzione attraverso una relazione con cardinalità (0,1) per Cliente e (1,1) per Online, non vi sarebbero sprechi di memoria dovuti a possibili valori null sui 3 attributi, ma solamente l'introduzione di un campo con chiave esterna verso Cliente per ogni cliente online.
Infine per quanto riguarda l'entità Prenotazione è stato scelto di eliminare la generalizzazione introducendo solamente un nuovo attributo “Confermata” che potrà assumere valori booleani. Non essendovi relazioni che coinvolgono direttamente i figli e non avendo questi attributi propri, è sufficiente distinguere per ogni prenotazione in che caso ci si trovi.
Si valutano a questo punto accorpamenti e partizionamenti di entità o relazioni. L'entità Accessorio potrebbe essere partizionata in cabine, lettini e sedie. Dato che ad ogni prenotazione potrebbero essere associati indistintamente accessori disomogenei, risulterebbe scomodo cercare ogni volta in tabelle distinte. Inoltre bisognerebbe indicare nella relazione PAccessorio dove cercare attraverso attributi aggiuntivi. Pertanto si è scelto di mantenere accorpati tali enti nell'entità dello schema E-R precedente. L'orario d'uso dei pedalò potrà essere inserito in un'unica tabella corrispondente a UsoPedalo, che avrà per campi un riferimento al cliente, uno al pedalò e l'orario appunto.
Infine si considerano gli identificatori principali. Dato che più clienti potrebbero avere nome e cognome uguali, viene mantenuta come chiave una matricola assegnata dal sistema informatico. Le

4 Stimata su 3 anni di utilizzo

prenotazioni potrebbero essere identificate dalla data d'inizio e l'ombrellone, ma siccome compaiono in molteplici relazioni, è stato introdotto un attributo ID che da solo le identifica, semplificando i riferimenti.
Le altre entità mantengono gli identificatori specificati dallo schema E-R originale.
Applicando le modifiche allo schema E-R, traducendolo in schema logico e controllando le forme normali (come si vedrà in seguito), si sono trovati i seguenti schemi di relazione[5]:



Forme normali
Analizzando le dipendenze funzionali (FD) delle relazioni com'erano nel primo momento successivo alla loro creazione (dallo schema E-R), si era notato che
Accessorio(Matricola,Tipo,Prezzo) e
Ombrellone(Fila,Numero,Fascia)
violavano la forma normale di Boyce e Codd (BCNF). In Accessorio era infatti presente la FD Tipo → Prezzo, dove Tipo non è superchiave, ed in Ombrellone la FD Fila → Fascia, con Fila che non contiene una chiave della relazione. Per questo motivo le due relazioni sono state decomposte nella forma visibile nella rappresentazione precedente. Le decomposizioni sono entrambe senza perdita e conservano le dipendenze; sono inoltre in BCNF. A questo punto tutte le relazioni sono in BCNF.

5 Le chiavi esterne sono state evidenziate attraverso le frecce tratteggiate; le chiavi primarie dalla parola key accanto all'attributo (se vi sono più attributi contrassegnati da key, la chiave è formata dall'insieme che li contiene tutti).

IMPLEMENTAZIONE

Lo schema logico precedente è stato implementato nel DBMS Mysql nel modo seguente:

```
set FOREIGN_KEY_CHECKS=0;

create table if not exists Cliente(
Matricola int primary key,
Nome varchar(25) not null,
Cognome varchar(25) not null
)Engine=InnoDB;

create table if not exists Online(
Cliente int primary key,
User varchar(30) unique not null,
Password varchar(25) not null,
Mail varchar(100) not null,
FOREIGN KEY (Cliente)
REFERENCES Cliente(Matricola)
on delete cascade
)Engine=InnoDB;

create table if not exists Ombrellone(
Fila char(1),
Numero smallint,
primary key (Fila,Numero)
)Engine=InnoDB;

create table if not exists Fascia(
Fila char(1) primary key,
Fascia smallint not null,
FOREIGN KEY (Fila)
 REFERENCES Ombrellone(Fila)
)Engine=InnoDB;

create table if not exists Prezzo(
Fascia smallint,
Anno numeric(4),
Mese numeric(2),
Prezzo numeric(6,2) not null,
primary key (Fascia,Anno,Mese)
/*FOREIGN KEY (Fascia)
REFERENCES Fascia(Fascia)
on delete cascade*/
)Engine=InnoDB;


create table if not exists Accessorio(
Matricola char(4) primary key,
Tipo varchar(10) not null
)Engine=InnoDB;

create table if not exists CostoACC(
Tipo varchar(10) primary key,
Prezzo numeric(6,2) not null
/*FOREIGN KEY (Tipo)
 REFERENCES Accessorio(Tipo)
on delete cascade*/
)Engine=InnoDB;

create table if not exists Prenotazione(
ID int primary key,
Cliente int not null,
FilaOmbrellone char(1) not null,
NumeroOmbrellone smallint not null,
Inizio date not null,
Fine date not null,
Creazione datetime not null /*default NOW()*/,
Confermata tinyint(1) not null,
FOREIGN KEY (Cliente)
 REFERENCES Cliente(Matricola),
FOREIGN KEY (FilaOmbrellone,NumeroOmbrellone)
 REFERENCES Ombrellone(Fila,Numero)
/*unique (FilaOmbrellone,NumeroOmbrellone,Inizio);*/
)Engine=InnoDB;

create table if not exists Pagamento(
Fattura int primary key,
Importo numeric(7,2) not null,
Data date not null,
Cliente int,
Tipo varchar(20),
FOREIGN KEY (Cliente) REFERENCES Cliente(Matricola)
)Engine=InnoDB;

create table if not exists PAccessorio(
Prenotazione int,
Accessorio char(4),
Inizio date not null,
Fine date not null,
Pagato smallint default 0,
primary key(Prenotazione,Accessorio),
FOREIGN KEY (Prenotazione) REFERENCES Prenotazione(ID) on
delete cascade,
FOREIGN KEY (Accessorio) REFERENCES Accessorio(Matricola)
)Engine=InnoDB;

create table if not exists Pedalo(
Matricola char(3) primary key,
Posti smallint,
Costo numeric(6,2) not null
)Engine=InnoDB;

create table if not exists UsoPedalo(
Pedalo char(3),
Data date,
Ora time,
Consegna time,
Cliente int not null,
Pagato smallint,
primary key (Pedalo,Data,Ora),
FOREIGN KEY (Pedalo) REFERENCES Pedalo(Matricola),
FOREIGN KEY (Cliente) REFERENCES Cliente(Matricola)
)Engine=InnoDB;

create table if not exists Dipendente(
Matricola int primary key,
Nome varchar(25) not null,
Cognome varchar(25) not null,
User varchar(30) unique not null,
Password varchar(25) not null,
Ruolo varchar(25)
)Engine=InnoDB;


set FOREIGN_KEY_CHECKS=1;
```

QUERY E VIEW RILEVANTI

Query che stampa per ogni ombrellone la fila, numero e fascia che lo caratterizzano, e un booleano pari a 1 se e solo se l'ombrellone non è mai occupato da altre prenotazioni durante il periodo di tempo che va da $inizio a $fine.

```
select o.*,f.fascia,IF((o.fila,o.numero) in (
        select om2.*    from Ombrellone as om2, Prenotazione as pre
        where om2.Fila=pre.FilaOmbrellone and om2.Numero=pre.NumeroOmbrellone
        and (("$inizio">=pre.Inizio and "$inizio"<=pre.Fine) or
        ("$fine">=pre.Inizio and "$fine"<=pre.Fine) or ("$inizio"<=pre.Inizio and
        "$fine">=pre.Fine))
),0,1) as disponibile from Ombrellone as o join Fascia as f on o.fila=f.fila;
```

View che mostra per ogni prenotazione gli accessori prenotati, con relativo intervallo temporale (se non vi fossero accessori associati allora viene mostrato Null).

```
create view distribuzioneAccessori as
select p.ID,pa.Accessorio,pa.inizio,pa.fine,a.tipo
from Prenotazione as p left join PAccessorio as pa on p.ID=pa.Prenotazione left join Accessorio as a
on pa.accessorio=a.matricola;
```

Query che mostra tutti gli accessori prenotati per la prenotazione avente ID $id. In particolare si vogliono raggruppare gli accessori dello stesso tipo che sono prenotati per lo stesso intervallo temporale, e stampare quindi il tipo di accessorio, le date e il numero di accessori (di quel determinato tipo prenotati per quelle date).

```
select ID,tipo,inizio,fine,count(tipo) as numero
from distribuzioneAccessori where id="$id"
group by ID,tipo,inizio,fine;
```

Query che mostri gli ombrelloni attualmente in uso, mostrandone fila e numero. I risultati devono essere ordinati per fila e numero.

```
select o.* from Ombrellone as o join Prenotazione as p
on (o.fila,o.numero)=(p.filaombrellone,p.numeroombrellone)
where p.inizio<=CURDATE() and p.fine>=CURDATE() order by o.fila,o.numero;
```

Query che mostra la durata media annua delle prenotazioni (di ombrelloni) a partire da quest'anno e andando indietro.

```
select year(inizio) as anno,avg(datediff(fine,inizio)+1) as giorni
from Prenotazione group by anno order by anno desc;
```

Query che mostri la distribuzione odierna degli accessori sugli ombrelloni: per ogni ombrellone occupato, che abbia associato almeno un accessorio, si devono stampare fila e numero corrispondenti, e tipi di accessorio con relativo numero (di accessori di questo tipo). Si devono considerare solo gli accessori per cui si ha già pagato. Si chiede inoltre di ordinare i risultati per fila e numero di ombrellone e tipo di accessorio.

Per realizzare questa query si è fatto uso della seguente vista:

```
create view distribuzioneAccessori2 as
select p.ID,pa.Accessorio,pa.inizio,pa.fine,a.tipo,pa.pagato
from Prenotazione as p left join PAccessorio as pa on p.ID=pa.Prenotazione left join Accessorio as a
on pa.accessorio=a.matricola;

select p.filaombrellone,p.numeroombrellone,d.tipo,count(d.tipo)
from distribuzioneAccessori2 as d join Prenotazione as p on d.id=p.id
where (d.tipo='Lettino' or d.tipo='Sedia') and d.pagato='1' and d.inizio<=curdate()
and d.fine>=curdate() group by d.tipo,p.filaombrellone,p.numeroombrellone
order by filaombrellone,numeroombrellone,tipo;
```

FUNZIONI E PROCEDURE RILEVANTI

Funzione che calcola il costo per l'ombrellone di una prenotazione, in base al prezzo giornaliero e al periodo coperto.

```
delimiter |
create function costoPre(pre int)
returns numeric(7,2)
BEGIN
declare importo numeric(7,2);
declare fattore numeric(7,2);
declare giorni smallint;
declare attuale date;
declare limite date;
declare terminazione date;
set importo=0.00;
select p.Inizio into attuale from Prenotazione as p where p.ID=pre;
select p.Fine into terminazione from Prenotazione as p where p.ID=pre;
REPEAT
        select last_day(attuale) into limite;
        if limite > terminazione then set limite=terminazione;
        end if;
        select datediff(limite,attuale) into giorni;
        set giorni=giorni+1;
        select co.Prezzo into fattore from Prenotazione as p,Fascia as fa,Prezzo as co
        where p.ID=pre and p.FilaOmbrellone=fa.Fila and co.Fascia=fa.Fascia and
co.Anno=year(attuale)
        and  co.Mese=month(attuale);
set importo=importo+fattore*giorni;
set attuale=limite;
set attuale=date_add(attuale,interval 1 day);
UNTIL attuale > terminazione END REPEAT;
return importo; end |
delimiter ;
```

Funzione che restituisce il numero di accessori, del tipo richiesto, che sono disponibili per un intervallo indicato.

```
delimiter |
create function numAcc(inizio date,fine date,acc varchar(10))
returns int
begin
declare n int;
select count(a.Matricola) into n
from Accessorio as a
where a.Tipo=acc and a.Matricola not in
(select a2.Matricola
from Accessorio as a2 join PAccessorio as pre on a2.Matricola=pre.Accessorio
where (inizio>=pre.Inizio and inizio<=pre.Fine) or
        (fine>=pre.Inizio and fine<=pre.Fine) or (inizio<=pre.Inizio and fine>=pre.Fine));
return n;
end |
delimiter ;
```

Procedura che memorizza una prenotazione di accessorio, dopo aver controllato la disponibilità di questo nelle date indicate.

```
delimiter |
create procedure preAcc(inizio date,fine date,tipo varchar(10),id int,pagato smallint)
begin
declare n int;
declare acc char(4);
declare ini date;
declare fin date;
select p.inizio,p.fine into ini,fin from Prenotazione as p where p.id=id;
if inizio>=ini and fine<=fin then
select numAcc(inizio,fine,tipo) into n;
if n>0 then
select min(a.Matricola) into acc
from Accessorio as a
where a.Tipo=tipo and a.Matricola not in
(select a2.Matricola
from Accessorio as a2 join PAccessorio as pre on a2.Matricola=pre.Accessorio
where (inizio>=pre.Inizio and inizio<=pre.Fine) or
        (fine>=pre.Inizio and fine<=pre.Fine) or (inizio<=pre.Inizio and fine>=pre.Fine));
insert into PAccessorio values(id,acc,inizio,fine,pagato);
end if; end if; end |
delimiter ;
```

Funzione che calcola il prezzo totale relativo alla prenotazione di tutti e soli gli accessori associati ad una prenotazione di ombrellone.

```
delimiter |
create function costoAcc(id int)
returns numeric(7,2)
begin
DECLARE done INT DEFAULT 0;
declare giorni smallint;
declare acc char(4);
declare costo numeric(7,2);
declare tot numeric(7,2) default 0;
declare cur CURSOR for select accessorio,datediff(fine,inizio) from PAccessorio where
Prenotazione=id;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
set tot=0;
open cur;
repeat
        set costo=0;
        set giorni=0;
        fetch cur into acc,giorni;
        if not done then
        set giorni=giorni+1;
        select ca.prezzo into costo from CostoACC as ca
        where ca.tipo=(select a.tipo from Accessorio as a where a.Matricola=acc);
        set tot=tot+(giorni*costo);
        end if;
until done end repeat;
close cur;
return tot;
end |
delimiter ;
```

Procedura che inserisce un nuovo record nella tabella Prezzo se viene passato il prezzo relativo ad un periodo non ancora presente nella tabella, e che invece aggiorna la tabella se il prezzo indicato è relativo ad un periodo già presente.

```
delimiter |
create procedure fissaPrezzo(ann int,mes int,fas smallint,pre numeric(7,2))
begin
declare pres smallint;
select (ann,mes,fas) in (select p.anno,p.mese,p.fascia from Prezzo as p) into pres;
if pres then
update Prezzo set prezzo=pre where anno=ann and mese=mes and fascia=fas;
else
insert into Prezzo values(fas,ann,mes,pre);
end if;
end|
delimiter ;
```

TRIGGER RILEVANTI

Trigger che ad ogni inserimento di prenotazione (di ombrellone) controlla non vi siano sovrapposizioni; in tal caso annulla l'inserimento.

```
delimiter |
create trigger controlloDate
before insert on Prenotazione
for each row
begin
declare errore smallint;
select count(*) into errore from Prenotazione as p
where p.FilaOmbrellone=NEW.FilaOmbrellone and
p.NumeroOmbrellone=new.NumeroOmbrellone
and ((new.inizio>=p.inizio and new.inizio<=p.fine)or(new.fine>=p.inizio and new.fine<=p.fine)
 or (new.inizio<=p.inizio and new.fine>=p.fine));
if errore>0 then
insert into Prenotazione select * from Prenotazione limit 1;
end if;
end |
delimiter ;
```

Trigger che registra una transazione di denaro nella relazione Pagamento ogni volta che una prenotazione (di ombrellone) è segnata come “Confermata”.

```
delimiter |
create trigger fatturaPagamento
after update on Prenotazione
for each row
begin
declare importo numeric(7,2);
declare fatt int;
declare giorni smallint;
if old.confermata=0 and new.confermata=1 then
select datediff(new.fine,new.inizio) into giorni;
set giorni=giorni+1;
select costoPre(new.ID) into importo;
select if(isnull(max(fattura)),0,max(fattura)) into fatt from Pagamento;
set fatt=fatt+1;
insert into Pagamento values(fatt,importo,CURDATE(),new.Cliente,'Ombrellone');
end if;
end |
delimiter ;
```

Trigger che registra una transazione di denaro nella relazione Pagamento ogni volta che un accessorio prenotato è segnato come “Pagato”.

```
delimiter |
create trigger fatturaAccessorio
after update on Paccessorio for each row
begin
declare importo numeric(7,2);
declare fatt int;
declare giorni smallint;
declare tipo varchar(20);
declare cliente int;
if old.pagato=0 and new.pagato=1 then
select datediff(new.fine,new.inizio) into giorni;
set giorni=giorni+1;
select c.prezzo into importo from PAccessorio as p join Accessorio as a on
p.Accessorio=a.matricola
join CostoACC as c on a.tipo=c.tipo where p.prenotazione=new.prenotazione and
p.accessorio=new.accessorio;
set importo=importo*giorni;
select if(isnull(max(fattura)),0,max(fattura)) into fatt from Pagamento;
set fatt=fatt+1;
select a.tipo into tipo from PAccessorio as p join Accessorio as a on p.accessorio=a.matricola
where p.prenotazione=new.prenotazione and p.accessorio=new.accessorio;
select p.cliente into cliente from Prenotazione as p where p.id=new.prenotazione;
insert into Pagamento values(fatt,importo,CURDATE(),cliente,tipo);
end if; end |
delimiter ;
```

ULTIMI DETTAGLI IMPLEMENTATIVI

1. Dato che (per scelta dello stabilimento) non si vuole richiedere il codice fiscale ad ogni cliente, viene usata una matricola per identificare univocamente un cliente. Casi di omonimia possono pertanto verificarsi ed essere gestiti correttamente. Nella pagina “dipendente.php” tuttavia è stato necessario ricavare i dati di un cliente dal nome e dal periodo di prenotazione, facendo implicitamente l'assunzione che non vi siano omonimi aventi prenotato un ombrellone nello stesso periodo di tempo (stesso nome, stesso cognome, stessa data d'inizio e stessa data di fine).

2. In parti cruciali del sito, come la memorizzazione nel database di una prenotazione, sono state usate delle transaction con eventuale roll back in caso di errore in una delle varie query invocate. Ad esempio se vengono prenotati 2 ombrelloni ed uno fosse in realtà non disponibile, il trigger “controlloDate” causerebbe un errore in una query e la transaction annullerebbe l'intera prenotazione.

3. Il pagamento di una prenotazione online causa un problema: da un lato si vorrebbe dare del tempo al cliente per pensare prima di pagare, e pertanto bisogna che in questo tempo l'ombrellone selezionato venga mantenuto “riservato” per non essere preso da altri clienti, evitando creare race conditions. Ma dall'altro lato un ombrellone non può rimanere riservato a lungo, col rischio di rimanere inutilizzato se poi il cliente decidesse di annullare e/o non pagare.
La soluzione trovata prevede la memorizzazione nel database dell'ora precisa di quando una prenotazione è stata fatta e, se dopo 5 minuti da questo tempo non vi è conferma (e quindi pagamento), la prenotazione viene cancellata automaticamente. Per fare ciò si era creato il seguente evento, che ogni minuto cancella tutte le prenotazioni vecchie di 5 minuti, che non sono state confermate:

```
SET GLOBAL event_scheduler = 0;
drop event if exists convalida
create event convalida
ON SCHEDULE
    EVERY 1 minute
do delete from Prenotazione where confermata=0 and creazione<(now()- interval 5 minute);
SET GLOBAL event_scheduler = 1;
```

Tuttavia, non disponendo nel DBMS dei permessi necessari per mandare l'evento in esecuzione, è stata trovata una seconda soluzione che sfrutta il server php. Ogni volta che viene caricata la pagina “funzioni.php”, che è in realtà richiamata da ogni pagina del sito, viene invocata la seguente query di delete:

```
delete from Prenotazione where confermata=0 and creazione<(now()- interval 5 minute);
```

l'effetto causato è sostanzialmente equivalente, con la sola differenza che l'evento non è più interno al database, come sarebbe stato nella prima soluzione, ma è ora gestito dal server php.