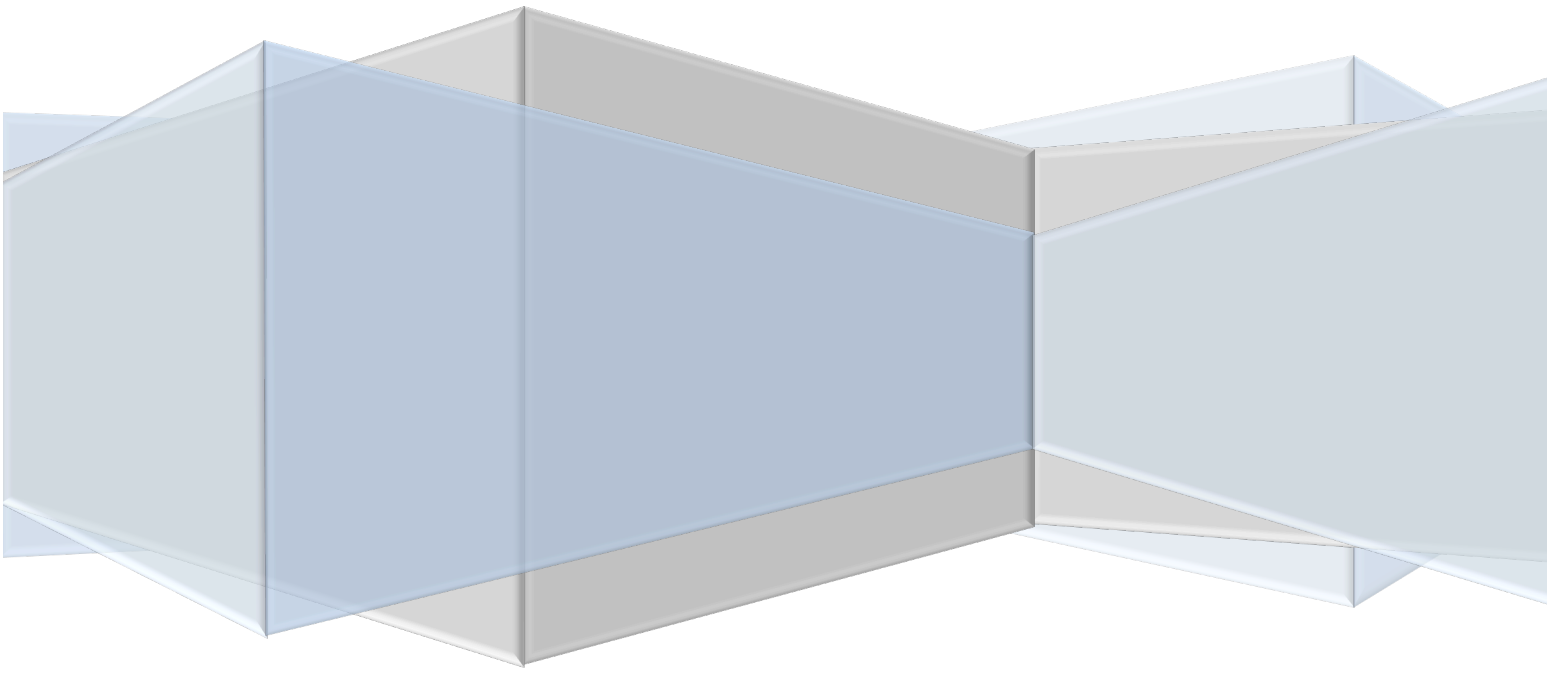
Anno Accademico 2013-2014

Presentazione del Progetto di Basi di Dati

Gestione di una piscina comunale

Autori

Nome	Cognome	Matricola
Alberto	Ferrara	1049378
Franco	Berton	1052574



Sommario

Abstract

Tale progetto consiste nella realizzazione di una base di dati per la gestione di una piscina comunale.
La clientela avrà la possibilità di svolgere una delle seguenti attività: acquagym, acquatherapy, aquabuilding, nuoto. Gli utenti potranno sottoscrivere un abbonamento, che darà il diritto di seguire uno qualsiasi di questi corsi, oppure potranno usufruire dei servizi offerti in giornata (senza quindi abbonarsi). Ogni attività sarà coordinata da un istruttore e si terrà in una piscina sorvegliata da un bagnino. Le operazioni tipiche sono aggiunta e modifica dei clienti, modifica dei prezzi , aggiunta di nuove strutture.

Analisi dei Requisiti

Il progetto consiste nella realizzazione di un database che modelli alcune classi riguardanti la gestione di una piscina comunale, il personale dedicato alla gestione della clientela e della struttura ha accesso alla base di dati mediante un'interfaccia web.
Ora, indicheremo le entità progettuali e le relative informazioni necessarie:
Delle **attività** ci interessa il giorno in cui sono svolte (L,Ma,Me,G,V,S,D), il tipo di attività e l'età minima per l'iscrizione.
Dei **clienti** che andranno a svolgere le attività ci interessa il nome, il cognome, la data di nascita, il luogo di nascita, il sesso e il codice fiscale. I clienti possono essere **abbonati** oppure **occasionali**, nel primo caso ci interesserà la data di inizio e di fine dell'abbonamento, il numero di badge e che tipo di abbonamento hanno sottoscritto. Nel secondo caso ci interessa la data e l'orario di entrata e infine il prezzo pagato. Tutti i **tipi di abbonamento** sono descritti in una classe contenente il prezzo e la durata per ogni abbonamento, mentre per le entrate occasionali avremo una classe **prezzi entrate** contenente il giorno (feriale/festivo), il tipo di entrata (giornaliera, pomeridiana o mattutina) e il prezzo. Ogni abbonato sarà titolare di un **badge** identificato mediante un apposito codice e caratterizzato da una data di scadenza. Le attività si svolgeranno nelle **piscine** che saranno caratterizzate dal nome, dalla lunghezza, profondità e larghezza, ci servirà poi sapere se è riscaldata o no, l'allocazione (interna o esterna) e il periodo di apertura (estivo o annuale).
Tutto questo sarà gestito dal **personale**, del quale ci interessa il nome, cognome, data e luogo di nascita, indirizzo, sesso, codice fiscale, recapito telefonico e

retribuzione. Il personale viene distinto in **segretari** del quale ci interessa il ruolo e la password (utilizzata per accedere alla base di dati) e in **staff tecnico**, del quale invece ci interessa la data di rilascio del brevetto, la data di fine validità di quest'ultimo e il numero identificativo del brevetto. Lo staff tecnico può essere **bagnino** o **istruttore**.
Solo i segretari hanno il diritto di inserire nuovi clienti e di eseguire tutte le operazioni possibili sulla base di dati.

Progettazione Concettuale

Lista delle classi

Classi	Attributi
Attività	Giorno: enum Tipo: enum Etàminima: int
Piscine	Nome: string Larghezza: int Lunghezza: int Profondità: int Riscaldata: bool Allocazione: enum Periodo Apertura: enum
Personale	Nome: string Cognome: string Data di nascita: date Luogo nascita: string Indirizzo: string Sesso: enum('M,'F') Codice fiscale: string Recapito Telefonico: int Retribuzione: real
Staff Tecnico	BrevettoRilasciatoil: date Validitàfinoa: date NumeroBrevetto: int
Sottoclassi	**Istruttori**
	Bagnini

Clienti		Nome: string Cognome: string Datanascita: date LuogoNascita: string Sesso: enum CodiceFiscale: string
Sottoclassi	**Abbonati**	DataInizio: date DataFine: date
	Occasionali	DataOrarioentrata: datetime
Prezzi Entrate		Giorno: enum TipoEntrata: enum Prezzo: real
Abbonamenti		Prezzo: real Durata: enum
Badge		CodiceTessera: int DataScadenza: date

Lista delle Associazioni

➢ **Clienti-Attività:** *Svolgono*

- Ogni cliente svolge 1 o più attività, ogni attività è effettuata da 0 o più clienti
- Molteplicità N:M
- Totalità: totale verso attività, parziale verso clienti

➢ **Attività-Piscine**: *Avvengono in*

- Ogni attività avviene in una piscina e ogni piscina ospita 0 o più attività
- Molteplicità N:1
- Totalità: totale verso piscine, parziale verso attività

➢ **Badge-Abbonati**: *Appartiene a*

- Ogni badge appartiene ad un abbonato o ogni abbonato possiede un badge
- Molteplicità: 1:1
- Totalità: totale in entrambi i versi

➢ **Abbonati-Abbonamenti**: *Titolare di*

- Ogni abbonato è titolare di un abbonamento e ogni abbonamento può essere sottoscritto da 0 o più abbonati
- Molteplicità: N:1
- Totalità: totale abbonamenti, parziale verso abbonati

➢ **Occasionali-PrezziEntrate**: *Pagano*

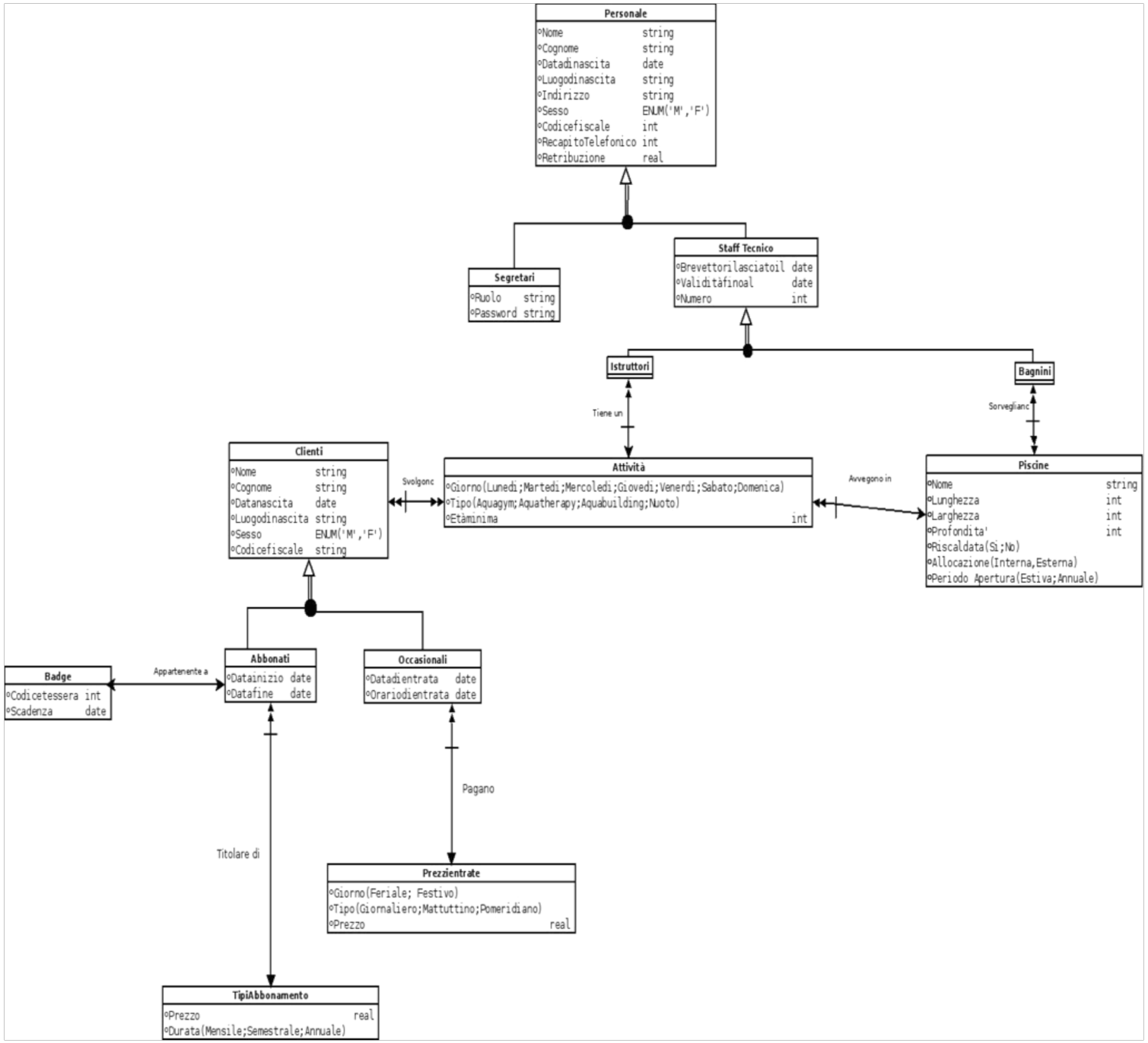
- Ogni utente occasionale paga un prezzo per entrare e ogni prezzo è pagato da 0 o più utenti occasionali
- Molteplicità N:1
- Totalità: totale verso PrezziEntrate, parziale verso Occasionali

- **Istruttori-Attività:** *Tengono*

- Ogni istruttore tiene un'attività e ogni attività è tenuta da 1 o più istruttori
- Molteplicità: N:1
- Totalità: totale verso istruttori, parziale verso Attività

- **Bagnini-Piscine:** *Sorvegliano*

- Ogni bagnino sorveglia 0 o più piscine o ogni piscina è sorvegliata da uno o più bagnini
- Molteplicità: N:M
- Totalità: totale verso Bagnini, parziale verso Piscine


Personale
°Nome string
°Cognome string
°Datadinascita date
°Luogodinascita string
°Indirizzo string
°Sesso ENUM('M','F')
°Codicefiscale int
°RecapitoTelefonico int
°Retribuzione real
Segretari
°Ruolo string
°Password string
Staff Tecnico
°Brevettorilasciatoil date
°Validitàfinoal date
°Numero int
Istruttori
Bagnini
Tiene un
Sorveglianc
Clienti
°Nome string
°Cognome string
°Datanascita date
°Luogodinascita string
°Sesso ENUM('M','F')
°Codicefiscale string
Svolgonc
Attività
°Giorno(Lunedì;Martedì;Mercoledì;Giovedì;Venerdì;Sabato;Domenica)
°Tipo(Aquagym;Aquatherapy;Aquabuilding;Nuoto)
°Etàminima int
Avvegono in
Piscine
°Nome string
°Lunghezza int
°Larghezza int
°Profondita' int
°Riscaldata(Sì;No)
°Allocazione(Interna,Esterna)
°Periodo Apertura(Estiva;Annuale)
Badge
°Codicetessera int
°Scadenza date
Appartenente a
Abbonati
°Datainizio date
°Datafine date
Occasionali
°Datadientrata date
°Orariodientrata date
Pagano
Titolare di
Prezzientrate
°Giorno(Feriale; Festivo)
°Tipo(Giornaliero;Mattuttino;Pomeridiano)
°Prezzo real
TipiAbbonamento
°Prezzo real
°Durata(Mensile;Semestrale;Annuale)

Progettazione Logica

Chiavi sintetiche

In ogni classe sono state aggiunte delle chiavi sintetiche intere (CodiceID, Codcliente, CodiceAbb, CodPrezzo, CodAttività, CodiceIstruttore, CodiceBagnino, CodicePiscina e CodOccasionale come chiave primaria perche CodiceID di Occasionali potrebbe essere ripetuto se un cliente occasionale viene piu di una volta) perché, oltre per un fattore di chiarezza, essendo il valore nullo un possibile valore di quasi tutti i campi, si è preferito assicurarsi che la chiave primaria non fosse inconsistente.

Gerarchie

La gerarchia clienti è stata implementata con partizionamento verticale visto che le sottoclassi abbonati e occasionali hanno alcuni attributi autonomi. La gerarchia completa è quindi implementata con le seguenti classi:

- **Clienti:**
  - CodCliente: int <<PK>>
  - Nome: string <<NOT NULL>>
  - Cognome: string <<NOT NULL>>
  - DatadiNascita: date
  - LuogodiNascita: string
  - Sesso: enum('M','F')
  - CodiceFiscale: string

- **Abbonati:**
  - CodCliente: int <<PK>> <<FK(Clienti)>>
  - DataInizio: date <<NOT NULL>>
  - Datafine: date <<NOT NULL>>
  - Badge: int
  - CodAbb: int

- **Occasionali:**
  - CodOccasionale: int <<PK>>
  - CodCliente: int <<FK(Clienti)>>
  - DataOrariodientrata: datetime <<NOT NULL>>
  - Prezzo: int <<FK(PrezziEntrate)>>

La gerarchia personale è stata implementata con partizionamento verticale poiché la sottoclasse segretari ha come scopo quello di memorizzare gli utenti che possono accedere e modificare la base di dati ma che non hanno nessun altro collegamento con le altre classi del database. La sottoclasse staff tecnico della gerarchia personale è stata invece implementata con partizionamento orizzontale. La gerarchia completa è quindi implementata con le seguenti classi:

- **Personale:**
  - CodiceID: int <<PK>>
  - Nome: string <<NOT NULL>>
  - Cognome: string <<NOT NULL>>
  - DatadiNascita: date
  - LuogodiNascita: string
  - Indirizzo: string
  - Sesso: enum('M','F')
  - CodiceFiscale: string
  - RecapitoTelefonico: int
  - Retribuzione: real

- **Segretari**
  - CodiceID: int <<PK>> <<FK(Personale)>>
  - Ruolo: string <<NOT NULL>>
  - Password: string <<NOT NULL>>

- **Istruttori**
  - CodiceID: int <<PK>> <<FK(Personale)>>
  - Brevettorilasciatoil: date
  - ScadenzaBrevetto: date
  - NumeroBrevetto: int <<NOT NULL>>

- **Bagnini**
  - CodiceID: int <<PK>> <<FK(Personale)>>
  - Brevettorilasciatoil: date
  - ScadenzaBrevetto: date
  - NumeroBrevetto: int <<NOT NULL>>

Lista delle associazioni

- **Abbonati-TipoAbbonamento:** *è titolare di*
  - Ogni Abbonato ha un solo tipo di abbonamento e ogni abbonamento può corrispondere a 0 o più abbonati
  - Molteplicità: 1:N
  - Totalità: totale verso TipoAbbonamento, parziale verso Abbonati
  - Chiave esterna non nulla in Abbonati verso TipoAbbonamento

- **Badge-Abbonati:** *Appartiene a*
  - Ogni badge appartiene ad un abbonato ed ogni abbonato è proprietario di un solo badge
  - Molteplicità: 1:1
  - Totalità: totale in entrambi i versi
  - Chiave esterna non nulla in abbonati verso Badge

- **Occasionali-PrezziEntrate:** *Pagano un*
  - Ogni cliente occasionale paga un prezzo di entrata e ogni prezzo è pagato da 0 o più clienti occasionali
  - Molteplicità: 1:N
  - Totalità: totale verso occasionali, parziale verso PrezziEntrate
  - Chiave esterna in Occasionali verso PrezziEntrate

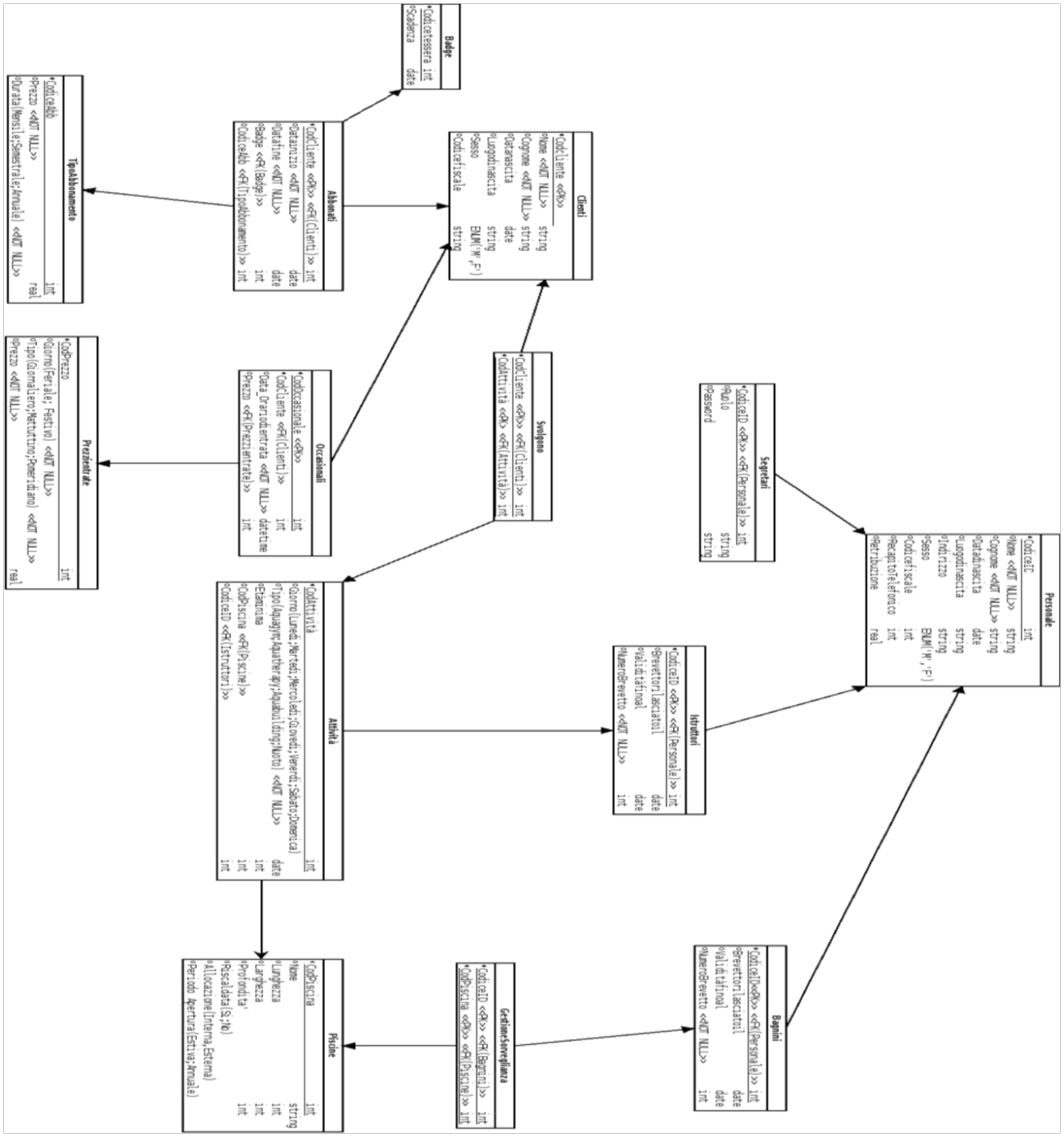
- **Istruttori-Attività:** *Tengono un*
  - Ogni istruttore tiene 0 o un'attività e ogni attività è tenuta da uno o più istruttori

- Molteplicita: N:1
- Totalità: totale verso Istruttori, parziale verso Attività
- Chiave esterna in Attività verso Istruttori

- **Attività-Piscine:** *Si tiene in*
  - Ogni attività avviene in una piscina e ogni piscina ospita 0 o più attivita
  - Molteplicità: N:1
  - Totalità: Totale verso piscine e parziale verso Attività
  - Chiave esterna in Attività verso Piscine

- **Clienti-Attività:** *Svolge un'*
  - Ogni cliente svolge una o più attività, e ogni attività è svolta da 0 o più clienti
  - Molteplicità N:M
  - Totalità: totale verso attività, parziale verso clienti
  - Nuova tabella **Svolgono** con i seguenti attributi:
    - CodCliente: int <<PK>> <<FK(Clienti)>>
    - CodAttività: <<PK>> <<FK(Attività)>>

- **Bagnini-Piscine:** *Sorvegliano*
  - Ogni bagnino sorveglia 0 o più piscine, ogni piscina è sorvegliata da 1 o più bagnini
  - Molteplicità N:M
  - Totalità: totale verso Bagnini, parziale verso Piscine
  - Nuova tabella **GestioneSorveglianza** con i seguenti attributi:
    - CodiceBagnino: int <<PK>> <<FK(Bagnini)>>
    - CodPiscina: int <<PK>> <<FK(Piscine)>>


Badge
*CodiceTessera int
#Scadenza date
Abbonati
*CodCliente <<PK>> <<FK(Clienti)>> int
#DataInizio <<NOT NULL>> date
#DataFine <<NOT NULL>> date
#Badge <<FK(Badge)>> int
#CodiceAbb <<FK(TipoAbbonamento)>> int
TipoAbbonamento
*CodiceAbb int
#Prezzo <<NOT NULL>> real
#Durata(Mensile;Semestrale;Annuale) <<NOT NULL>>
Clienti
*CodCliente <<PK>>
#Nome <<NOT NULL>> string
#Cognome <<NOT NULL>> string
#DataNascita date
#LuogoDiNascita string
#Sesso ENUM('M','F')
#CodiceFiscale string
Occasionali
*CodOccasionale <<PK>> int
*CodCliente <<FK(Clienti)>> int
#Data_OrarioEntrata <<NOT NULL>> datetime
#Prezzo <<FK(PrezziEntrate)>> int
PrezziEntrate
*CodPrezzo int
#Giorno(Feriale; Festivo) <<NOT NULL>>
#Tipo(Giornaliero;Mattutino;Pomeridiano) <<NOT NULL>>
#Prezzo <<NOT NULL>> real
Svolgono
*CodAttività <<PK>> <<FK(Attività)>> int
Attività
*CodAttività int
#Giorno(Lunedì;Martedì;Mercoledì;Giovedì;Venerdì;Sabato;Domenica)
#Tipo(Aquagym;AquaTherapy;AquaAudizione;Nuoto) <<NOT NULL>>
#Esaminata date
#CodPiscina <<FK(Piscine)>> int
#CodiceID <<FK(Istruttori)>> int
Piscine
*CodPiscina int
#Nome string
#Lunghezza int
#Larghezza int
#Profondità int
#Riscaldata(SI;No)
#Allocazione(Interna,Esterna)
#Periodo Apertura(Estiva;Annuale)
GestioneSorveglianza
*CodiceID <<PK>> <<FK(Bagnini)>> int
*CodPiscina <<PK>> <<FK(Piscine)>> int
Istruttori
*CodiceID <<PK>> <<FK(Personale)>> int
#BrevettoriLacustri date
#ValiditàTrimoli date
#NumeroBrevetto <<NOT NULL>> int
Bagnini
*CodiceIDnuoto <<PK>> <<FK(Personale)>> int
#BrevettoriLacustri date
#ValiditàTrimoli date
#NumeroBrevetto <<NOT NULL>> int
Segretari
*CodiceID <<PK>> <<FK(Personale)>> int
#Ruolo string
#Password string
Personale
*CodiceIC int
#Nome <<NOT NULL>> string
#Cognome <<NOT NULL>> string
#DataNascita date
#LuogoDiNascita string
#Indirizzo string
#Sesso ENUM('M','F')
#CodiceFiscale int
#RecapitoTelefonico int
#Retribuzione real

Implementazione della base di dati

```
SET FOREIGN_KEY_CHECKS=0;

USE Progetto;

DROP TABLE IF EXISTS Segretari;
DROP TABLE IF EXISTS GestioneSorveglianza;
DROP TABLE IF EXISTS Bagnini;
DROP TABLE IF EXISTS Piscine;
DROP TABLE IF EXISTS Istruttori;
DROP TABLE IF EXISTS Svolgono;
DROP TABLE IF EXISTS Attivita;
DROP TABLE IF EXISTS Abbonati;
DROP TABLE IF EXISTS Occasionali;
DROP TABLE IF EXISTS Clienti;
DROP TABLE IF EXISTS TipoAbbonamento;
DROP TABLE IF EXISTS PrezziEntrate;
DROP TABLE IF EXISTS Badge;
DROP TABLE IF EXISTS Personale;
DROP TABLE IF EXISTS Errori;



CREATE TABLE Personale (
    CodiceID      INT(6)      PRIMARY KEY AUTO_INCREMENT,
    Nome          VARCHAR(10)  NOT NULL,
    Cognome       VARCHAR(10)  NOT NULL,
    DataNascita   DATE,
    LuogoNascita  VARCHAR(20),
    Indirizzo     VARCHAR(30),
    Sesso         ENUM('M','F'),
    CodiceFiscale  VARCHAR(16),
    RecTelefonico  VARCHAR(11),
    Retribuzione   FLOAT(8)
    ) ENGINE=InnoDB;

CREATE TABLE Segretari (
```

```
    CodiceID      INT(6)      PRIMARY KEY,
    Ruolo         VARCHAR(15)  NOT NULL,
    Password      VARCHAR(15)  NOT NULL,
    CONSTRAINT FOREIGN KEY(CodiceID) REFERENCES Personale(CodiceID)
                          ON DELETE CASCADE
                          ON UPDATE CASCADE

    ) ENGINE=InnoDB;

CREATE TABLE Istruttori (
    CodiceID             INT(6)      PRIMARY KEY,
    Brevettorilasciatoil   DATE,
    ScadenzaBrevetto       DATE,
    NumeroBrevetto         INT(8)      NOT NULL,
    CONSTRAINT FOREIGN KEY (CodiceID) REFERENCES Personale(CodiceID)
                          ON DELETE CASCADE
                          ON UPDATE CASCADE

    ) ENGINE=InnoDB;

CREATE TABLE Bagnini (
    CodiceID             INT(6)      PRIMARY KEY,
    Brevettorilasciatoil   DATE,
    ScadenzaBrevetto       DATE,
    NumeroBrevetto         INT(8)      NOT NULL,
    CONSTRAINT FOREIGN KEY (CodiceID) REFERENCES Personale(CodiceID)
                          ON DELETE CASCADE
                          ON UPDATE CASCADE
    ) ENGINE=InnoDB;

CREATE TABLE Piscine (
    CodPiscina           INT(6)    PRIMARY KEY  AUTO_INCREMENT,
    Nome                VARCHAR(15),
    Lunghezza            FLOAT(8),
    Larghezza           FLOAT(8),
    Profondita           FLOAT(8),
    Riscaldata          ENUM('Si', 'No'),
    Allocazione          ENUM('Interna','Esterna'),
    PeriodoApertura      ENUM('Annuale','Estivo')
    ) ENGINE=InnoDB;
```

```
CREATE TABLE GestioneSorveglianza (
    CodiceID            INT(6),
    CodPiscina          INT(6),
    PRIMARY KEY (CodiceID, CodPiscina),
    CONSTRAINT FOREIGN KEY (CodiceID) REFERENCES Bagnini(CodiceID)
                        ON DELETE CASCADE
                        ON UPDATE CASCADE,
    CONSTRAINT FOREIGN KEY(CodPiscina) REFERENCES Piscine(CodPiscina)
                        ON DELETE CASCADE
                        ON UPDATE CASCADE
    ) ENGINE=InnoDB;

CREATE TABLE Attivita (
    CodAttivita         INT(6)     PRIMARY KEY AUTO_INCREMENT,
    Giorno
ENUM('Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'),
    Tipo              ENUM('Aquagym','Aquatherapy','Aquabuilding','Nuoto') NOT
NULL,
    Etaminima           INT(3),
    CodPiscina          INT(6),
    CodiceID            INT(6),
    CONSTRAINT FOREIGN KEY(CodPiscina) REFERENCES Piscine(CodPiscina)
                        ON DELETE CASCADE
                        ON UPDATE CASCADE,
    CONSTRAINT FOREIGN KEY(CodiceID) REFERENCES Istruttori(CodiceID)
                        ON DELETE CASCADE
                        ON UPDATE CASCADE
  ) ENGINE=InnoDB;

CREATE TABLE Clienti (
    CodCliente          INT(6)     PRIMARY KEY AUTO_INCREMENT,
    Nome              VARCHAR(15)  NOT NULL,
    Cognome             VARCHAR(15)  NOT NULL,
    DataNascita         DATE,
    LuogoNascita        VARCHAR(15),
    Sesso             ENUM('M','F'),
    CodiceFiscale       VARCHAR(16)
    ) ENGINE=InnoDB;

CREATE TABLE Svolgono (
    CodCliente          INT(6),
```

```
    CodAttivita          INT(6),
    PRIMARY KEY(CodCliente, CodAttivita),
    CONSTRAINT FOREIGN KEY (CodCliente) REFERENCES Clienti(CodCliente)
                          ON DELETE CASCADE
                          ON UPDATE CASCADE,
    CONSTRAINT FOREIGN KEY (CodAttivita) REFERENCES Attività(CodAttivita)
                          ON DELETE CASCADE
                          ON UPDATE CASCADE
    ) ENGINE=InnoDB;

CREATE TABLE TipoAbbonamento (
    CodiceAbb           INT(6)    PRIMARY KEY AUTO_INCREMENT,
    Prezzo              FLOAT(8)   NOT NULL,
    Durata              ENUM('Mensile','Semestrale','Annuale')   NOT NULL
    ) ENGINE=InnoDB;

CREATE TABLE PrezziEntrate (
    CodPrezzo           INT(6)    PRIMARY KEY AUTO_INCREMENT,
    Giorno              ENUM('Feriale','Festivo')   NOT NULL,
    Tipo                ENUM('Giornaliero','Pomeridiano','Mattutino')    NOT NULL,
    Prezzo              FLOAT(8)     NOT NULL
    ) ENGINE=InnoDB;

CREATE TABLE Badge (
    CodiceTessera       INT NOT NULL AUTO_INCREMENT,
    Datascadenza        DATE,
    PRIMARY KEY(CodiceTessera)
    ) ENGINE=InnoDB;

CREATE TABLE Abbonati (
    CodCliente          INT(6)   PRIMARY KEY,
    DataInizio          DATE     NOT NULL,
    DataFine            DATE     NOT NULL,
    Badge               INT(6)   NOT NULL AUTO_INCREMENT,
    CodiceAbb           INT(6),
    CONSTRAINT FOREIGN KEY (CodCliente) REFERENCES Clienti(CodCliente)
                          ON DELETE CASCADE
                          ON UPDATE CASCADE,
    CONSTRAINT FOREIGN KEY (Badge) REFERENCES  Badge(CodiceTessera)
                         ON DELETE CASCADE
                         ON UPDATE CASCADE,
```

```
    CONSTRAINT FOREIGN KEY (CodiceAbb) REFERENCES
TipoAbbonamento(CodiceAbb)
                            ON DELETE CASCADE
                            ON UPDATE CASCADE
    ) ENGINE=InnoDB;

CREATE TABLE Occasionali (
    CodOccasionale        INT(6) PRIMARY KEY AUTO_INCREMENT,
    CodCliente            INT(6),
    Data_ora_entrata      DATETIME  NOT NULL,
    Prezzo                INT(6),
    CONSTRAINT FOREIGN KEY (CodCliente) REFERENCES Clienti(CodCliente)
                            ON DELETE CASCADE
                            ON UPDATE CASCADE,
    CONSTRAINT FOREIGN KEY (Prezzo) REFERENCES PrezziEntrate(CodPrezzo)
                          ON DELETE CASCADE
                          ON UPDATE CASCADE
    ) ENGINE=InnoDB;


CREATE TABLE Errori (
    Coderr                INT(6) PRIMARY KEY AUTO_INCREMENT,
    Descrizione           VARCHAR(200)
    ) ENGINE=InnoDB;

SET FOREIGN_KEY_CHECKS=1;
```

Procedure/Funzioni e Trigger

1. Ad ogni inserimento nella tabella svolgono (i clienti svolgono le attività) viene controllato che l'età della persona iscritta sia >= dell'età richiesta per quel determinato corso e controlla che l'attività svolta da un cliente in un determinato giorno della settimana corrisponda ad una delle possibili attività disponibili quel giorno.

- Funzione che controlla il giorno

```
DROP PROCEDURE IF EXISTS ControlloGiorno;
DELIMITER $
CREATE PROCEDURE ControlloGiorno(IN NuovoCliente INT, AttivitaSvolta INT)
BEGIN
IF (SELECT COUNT(*)
    FROM Occasionali, Attivita
    WHERE Occasionali.Codcliente=NuovoCliente AND
Attivita.CodAttivita=AttivitaSvolta AND
DAYNAME(Occasionali.Data_ora_entrata)= Attivita.Giorno
 AND Occasionali.CodOccasionale= (SELECT MAX(Occasionali.CodOccasionale)
                    FROM Occasionali, Attivita
                    WHERE Occasionali.Codcliente=NuovoCliente AND
Attivita.CodAttivita=AttivitaSvolta))=0 THEN
 INSERT INTO Errori(Coderr) Values(3);
End IF;
END $
DELIMITER ;
```

- Funzione che controlla che sia rispettata l'età minima

```
DROP PROCEDURE IF EXISTS Controlloeta;
DELIMITER $
CREATE PROCEDURE Controlloeta(IN NuovoCliente INT, AttivitaSvolta INT)
BEGIN
```

```
DECLARE Etaminimatt INT(3);
DECLARE Datanascita DATE;
SELECT Attivita.Etaminima, Clienti.DataNascita INTO Etaminimatt, Datanascita
FROM Clienti , Attivita
WHERE Clienti.CodCliente = NuovoCliente AND
AttivitaSvolta=Attivita.CodAttivita;

IF ((DATEDIFF(CURRENT_DATE(),DataNascita))/365 < Etaminimatt)
THEN INSERT INTO Errori(Coderr) VALUES(1);
END IF;
END $
DELIMITER ;
```

- Trigger che utilizza le due funzioni appena descritte (*ControlloGiorno e ControlloEta)*

```
DROP TRIGGER IF EXISTS Controlli;
DELIMITER $
CREATE TRIGGER Controlli
BEFORE INSERT ON Svolgono
FOR EACH ROW
BEGIN
CALL Controlloeta(NEW.CodCliente,NEW.CodAttivita);
IF (Select COUNT(*) FROM Abbonati WHERE
Abbonati.Codcliente=New.CodCliente)=0 THEN
CALL ControlloGiorno(NEW.CodCliente,NEW.CodAttivita) ;
END IF ;
END $
DELIMITER ;
```

2. Ad ogni inserimento di Abbonati viene automaticamente inserito il badge e controllato che il tipo di abbonamento sottoscritto corrisponda alla differenza calcolata tra la data di fine e la data di inizio.

- Procedura definita per il controllo che il tipo di abbonamento sia corretto

```
DELIMITER $
DROP PROCEDURE IF EXISTS ControlloAbbonamento;

CREATE PROCEDURE ControlloAbbonamento(IN DataInizio DATE, DataFine DATE,
CodiceAbb INT)

BEGIN
DECLARE Mesi_calcolati INT(3);
DECLARE Durata1 VARCHAR(15);

SELECT DATEDIFF(DataFine, DataInizio)/30, TA.Durata INTO Mesi_calcolati, Durata1
FROM TipoAbbonamento TA
WHERE CodiceAbb=TA.CodiceAbb;
IF ((Mesi_calcolati=1 AND Durata1!='Mensile') ||
  (Mesi_calcolati=6 AND Durata1!='Semestrale') ||
  (Mesi_calcolati=12 AND Durata1!='Annuale'))

THEN INSERT INTO Errori(Coderr) VALUES(2);
END IF;
END $
DELIMITER ;
```

- Trigger per la creazione del Badge e che richiama la procedura appena definita (*ControlloAbbonamento*)

```
DELIMITER $
DROP TRIGGER IF EXISTS CreaBadge;

CREATE TRIGGER CreaBadge
BEFORE INSERT ON Abbonati
FOR EACH ROW
BEGIN

CALL ControlloAbbonamento(NEW.DataInizio, NEW.DataFine, NEW.CodiceAbb);
INSERT INTO Badge(Datascadenza) VALUES(CURRENT_DATE() + INTERVAL 3 YEAR);
```

```
END $
DELIMITER ;
```

3. Prima di cancellare una sorveglianza da gestione sorveglianza, viene controllato che vi sia un'altro o altri bagnini che sorvegliano la relativa piscina.

```
DROP TRIGGER IF EXISTS ControlloSorveglianza;
DELIMITER $
CREATE TRIGGER ControlloSorveglianza
BEFORE DELETE ON GestioneSorveglianza
FOR EACH ROW
BEGIN

IF(SELECT COUNT(*)AS NumeroSorveglianza
FROM  GestioneSorveglianza
WHERE GestioneSorveglianza.CodPiscina = OLD.CodPiscina)=1 THEN
INSERT INTO Errori(Coderr) VALUES(4);
END IF;
END $
DELIMITER ;
```

Query

1)Trovare la/le attività con il maggior numero di partecipanti:

```
DROP VIEW IF EXISTS NPartecipanti;

CREATE VIEW NPartecipanti (Attività, NumeroPartecipanti) AS
(SELECT CodAttività, COUNT(*) AS Partecipanti
FROM Svolgono
GROUP BY CodAttività);

SELECT Attività, NumeroPartecipanti
FROM NPartecipanti
WHERE NumeroPartecipanti=(SELECT MAX(NumeroPartecipanti)
                    FROM NPartecipanti);
```

OUTPUT:

```
+------------+--------------------------+
| Attività  | NumeroPartecipanti |
+------------+--------------------------+
|          1 |                        9 |
|         11 |                        9 |
+------------+--------------------------+
2 rows in set (0.00 sec)
```

2)Lista dei bagnini che sorvegliano le piscine dove avviene nuoto per ogni giorno della settimana

```
SELECT  P.Nome, P.Cognome, PS.Nome, A.Giorno
FROM Personale P NATURAL JOIN  GestioneSorveglianza GS JOIN Piscine PS ON
GS.CodPiscina=PS.CodPiscina JOIN Attività A ON A.Codpiscina=GS.CodPiscina
WHERE A.Tipo='Nuoto' ORDER BY PS.Nome, A.Giorno;
```

OUTPUT:

```
+----------+---------------+-----------+----------+
| Nome   | Cognome    | Nome    | Giorno  |
+----------+---------------+-----------+----------+
| Felice   | Centofanti | Piscina1 |      L     |
| Carla    | Pota          | Piscina1 |      L     |
| Felice   | Centofanti | Piscina1 |    Ma    |
| Carla    | Pota          | Piscina1 |    Ma    |
| Felice   | Centofanti | Piscina1 |    Me    |
| Carla    | Pota          | Piscina1 |    Me    |
| Felice   | Centofanti | Piscina1 |      G     |
| Carla    | Pota          | Piscina1 |      G     |
| Felice   | Centofanti | Piscina1 |      V     |
| Carla    | Pota          | Piscina1 |      V     |
| Carla    | Pota          | Piscina1 |      D     |
| Felice   | Centofanti | Piscina1 |      D     |
| Gloria   | Morandin   | Piscina2 |      S     |
| Carla    | Pota          | Piscina2 |      S     |
+----------+--------------+-------------+----------+
14 rows in set (0.00 sec)
```

3) Istruttori con brevetto che scade nell anno in corso

```
SELECT  p.Cognome, p.Nome
FROM Personale p NATURAL JOIN Istruttori i
WHERE YEAR(i.ScadenzaBrevetto)=year(CURRENT_DATE());
```

OUTPUT:

```
+--------------+------------+
| Cognome | Nome     |
+--------------+------------+
| Bello        | Marcello|
+--------------+------------+
1 row in set (0.00 sec)
```

4)Abbonati che abbiano sottoscritto un abbonamento semestrale e frequentanti della piscina comunale anche di domenica

```
SELECT distinct c.Cognome, c.Nome
FROM Clienti c NATURAL JOIN Abbonati a NATURAL JOIN tipoabbonamento t
NATURAL JOIN svolgono s NATURAL JOIN attività ai
WHERE t.Durata='Semestrale' AND ai.Giorno='D';
```

OUTPUT:

```
+----------------+------------+
| Cognome    | Nome    |
+----------------+------------+
| Ciardi         | Veronica|
| Paparesta  | Gianluca |
+---------------+-------------+
2 rows in set (0.00 sec)
```

5)Determinare le 3 Province che hanno maggior numero di clienti della piscina

```
SELECT c.LuogoNascita
FROM clienti c
GROUP BY c.LuogoNascita
ORDER BY COUNT(c.LuogoNascita) DESC
LIMIT 0,3;
```

OUTPUT:

```
+-------------------+
| LuogoNascita |
+-------------------+
| Padova           |
| Venezia          |
| Treviso          |
+-------------------+
3 rows in set (0.00 sec)
```

6)Determinare il numero di clienti seguiti da ogni istruttore

```
DROP VIEW IF EXISTS Clis;

CREATE VIEW Clis (Istruttore, Cliente) AS
SELECT  I.CodiceID, s.CodCliente
FROM Istruttori I LEFT JOIN (attività a JOIN svolgono s ON
s.CodAttività=a.CodAttività) ON I.CodiceID=a.CodiceID
GROUP BY I.CodiceID, s.CodCliente;

SELECT p.Nome,p.Cognome, COUNT (DISTINCT c.Cliente) AS ClientiSeguiti
FROM Clis c JOIN Personale p ON c.Istruttore=p.CodiceID
GROUP BY p.Nome,p.Cognome;
```

OUTPUT:

```
+-------------+-------------+------------------+
| Nome      | Cognome| ClientiSeguiti |
+-------------+-------------+------------------+
| Antonio  | Cesto      |               29 |
| Ernesto   | Lesto      |                5 |
| Eva         | Rossi      |               11 |
| Federica | Fontana   |               39 |
| Marcello | Bello      |               23 |
| Rossana  | Zero       |               20 |
| Samuele | Melli       |               18 |
+--------------+------------+------------------+
7 rows in set (0.00 sec)
```

7)Determinare il tipo di attività e il relativo numero dei clienti di sesso femminile che la svogle e che compieranno nell'anno corrente un età >= 20 e <= 30

```
SELECT a.Tipo, COUNT(*) AS Donne
FROM Clienti c NATURAL JOIN svolgono s NATURAL JOIN attività a
WHERE c.Sesso='F' AND YEAR(CURRENT_DATE())-YEAR(c.DataNascita)>=20 AND
YEAR(CURRENT_DATE())-YEAR(c.DataNascita)<=30
GROUP BY a.Tipo;
```

OUTPUT:

```
+------------------+----------+
| Tipo             | Donne |
+------------------+----------+
| Aquagym          |        4 |
| Aquatherapy      |        4 |
+------------------+----------+
2 rows in set (0.01 sec)
```

8)Numero di attività svolte in una piscina interna il giovedì che ha avuto almeno 5 partecipanti che sono maggiorenni o che compieranno la maggiore età nell'anno corrente

```
DROP VIEW IF EXISTS Attivitàgiovedi;

CREATE VIEW Attivitàgiovedi AS
SELECT a.Tipo, COUNT(DISTINCT(s.Codcliente)) AS Partecipanti
FROM svolgono s NATURAL JOIN clienti c JOIN attività a ON
s.Codattività=a.Codattività JOIN piscine p ON p.CodPiscina=a.CodPiscina
WHERE YEAR(CURRENT_DATE())-YEAR(c.DataNascita)>=18 AND
p.allocazione='Interna' AND a.Giorno='G'
GROUP BY a.Tipo;

SELECT Count(*) AS AttivitàSvolteGiovedì
FROM Attivitàgiovedi a
WHERE a.Partecipanti>=5;
```

OUTPUT:

```
+-----------------------------+
| AttivitàSvolteGiovedì   |
+-----------------------------+
|                           3 |
+-----------------------------+
1 row in set (0.00 sec)
```

9)Impostare che dopo il 5 Giugno di ogni anno l'attività di Acquatherapy si svolgerà all'esterno fino al 15 settembre

```
UPDATE Attività
SET Codpiscina=3
WHERE MONTH(CURRENT_DATE())>=6 AND
    DAY(CURRENT_DATE())>=5 AND
    MONTH(CURRENT_DATE())<9 AND
    Tipo='Acquatherapy';
```

Interfaccia web

L'interfaccia web consente di eseguire le operazioni fondamentali sulla base di dati. Queste operazioni sono: Inserimento (clienti, personale, attività e piscine), Aggiornamento (tutte le tabelle), Cancellazione (personale, clienti e attività) e Visualizzazione (tutte le tabelle).

La connessione alla base di dati avviene mediante l'inclusione del seguente file php che ha il compito di autenticarsi e selezionare il database che si vuole utilizzare.

Connessione_db.php:

```
<?php
$connessione=mysql_connect(basidati, 'aferrara', 'password_mysql')
or die($_SERVER['PHP_SELF'] . "Connessione fallita!");
mysql_select_db('aferrara', $connessione);
?>
```

La HomePage del progetto ha il compito di ricevere tramite form i campi dati 'Utente' e 'Password' per l'accesso alla base di dati. Gli utenti che hanno il permesso alla login sono solamente i segretari (tabella SQL) dato che sono unicamente loro che utilizzano questo gestionale. Per comodità è stato inserito un utente 'Admin' 'Password'.

Progetto.php

```
<HTML>
<head>
<style type="text/css">
h1{color:black;}  #benvenuti{position:absolute;
top:20%;
left:30%;
width:500px;
height:200px;
padding:0;
margin-left:-250px;
margin-top: -100px;}
body {background-image:url(swimming.jpg); text-align:center; }
p {color: black;}
```

```
@font-face {
font-family: 'BlackRose';
src: url('BLACKR~1.ttf') format('truetype');}
#login_form{
position: absolute;
top: 25%;
left: 67%;
bottom: 20%;
font-size: 18px;
text-align: center; }
</style>  </head>
<body style="margin: 0px;">
<div id="benvenuti" style="width: 40%; float: left;">
<h1> <font face="BlackRose" size=30>Benvenuti nel<br> Gestionale della
</font></h1>
<img src="piscina.png" />
</div>
<div id="login_form" width: 20% >
<form name="f1" method="POST" action="login.php" id="f1">
    <table >
    <h1> <font face="BlackRose" size=30> Accedi </font></h1>
         <tr>
             <td>
                 <input type="text"  name="username" autofocus required />
             </td>
        </tr>
        </tr>
             <td>
                 <input type="password" name="password" required />
             </td>
        </tr>
        <tr >
            <td>
                <input type="submit"  name="login" value="Login" style="font-
size:18px; " />
            </td>
       </tr>
</table>
 </form>
 </div>  </body>
</HTML>
```

La pagina Login ha il compito di controllare se i dati inseriti corrispondano con i dati presenti nella tabella Segretari (o se corrispondono a 'Admin' 'Password') e se questi corrispondono viene creata la sessione che è mantenuta per tutta la durata della navigazione (ogni pagina ha bisogno dell'autenticazione per essere utilizzata).

Login.php

```
<?php
//includo i file necessari a collegarmi al db con relativo script di accesso
include("connessione_db.php");
//variabili POST con
$username=$_POST["username"];
$password=$_POST["password"];
$query = "SELECT * FROM Segretari WHERE CodiceID = \"$username\" AND
password = \"$password\" ";
$ris = mysql_query($query, $connessione) or die (mysql_error());
$riga=mysql_fetch_array($ris);
//Prelevo l'identificativo dell'utente
$login=$riga['CodiceID'];

// Effettuo il controllo
IF ($login == NULL)
    $trovato = 0 ;
else $trovato = 1;
/* Username e password corrette */
IF($trovato === 1|| ($username=='admin' && $password=='password') ) {
IF($username=='admin' && $password=='password')   {
   $login='username';
   $password='password';}

 /*Registro la sessione*/
session_start();
/*Registro il codice dell'utente*/
$_SESSION['login'] = $login;
Header('Location: Homeamministrazione.php'); }
else  Header('Location: Progetto.php');
?>
```

Subito dopo l'autenticazione, si viene rimandati alla Homeamministrazione che è la pagina principale dalla quale si può scegliere l'operazione da effettuare.

Homeamministrazione.php

```
<?
/* attiva la sessione */
 session_start();
 /* verifica se la variabile 'login' e` settata */
$login=$_SESSION['login'];
IF (empty($login)) {
/* non passato per il login: accesso non autorizzato ! */
echo "Impossibile accedere. Prima effettuare il login."; }
ELSE {
?>
<HTML>
<HEAD>
<Style type="text/css">
BODY {background-image:url(swimming.jpg); text-align:center; }
@font-face {
font-family: 'BlackRose';
src: url('BLACKR~1.ttf') format('truetype');}
</Style>
</HEAD>
<BODY>
<font face="BlackRose" size=5>
<h1>   Amministrazione </h1>
<h2> Seleziona un'operazione </h2>
<table width="100%" CELLSPACING="50" align="center">
<tr><h3>       <td align="center" ><h1>

<a href="Inserimento.php">Inserimento</a>
</h1></td>    </tr>
<tr>
<td align="center" ><h1>

<a href="Aggiornamento.php">Aggiornamento</a>
</h1></td>    </tr>
<tr>
<td align="center" ><h1>
```

```
<a href="Cancellazione.php">Cancellazione</a>
</h1></td>    </tr>
 <tr>
<td align="center" ><h1>

<a href="Visualizzazione.php">Visualizzazione</a>
</h1></td>    </tr>
</table>
</font>
<a href="logout.php">LOGOUT</a></h2>
</BODY>
</HTML>
<?
}
?>
```

Una delle scelte previste dalla Homeamministrazione è quella di LOGOUT che rimanda alla pagina *logout* la quale distrugge la sessione corrente.

Logout.php

```
<?php
session_start();

session_destroy();
?>
Logout effettuato! <br />
<a href="Progetto.php"> Torna alla pagina di Login </a>
```

Per comodità e chiarezza il codice delle varie operazioni non è stato riportato. L'organizzazione delle pagine in linea generale è la seguente:

- INSERIMENTO: Ad ogni tipo di inserimento corrisponde una pagina con una form, nella quale inserire i dati che verranno *salvati* in variabili php da un'altra pagina (le pagine che sono state chiamate di 'recupero'). Queste, oltre a memorizzare le variabili, hanno il compito di effettuare l'inserimento (in questo caso) tramite una query da inviare al database mediante l'apposita

istruzione. Se ci fossero degli errori durante l'esecuzione della query, questi verrebbero mostrati in output.

- AGGIORNAMENTO: Ad ogni aggiornamento corrisponde una pagina nella quale si seleziona chi/cosa modificare (Attività, Piscina, Bagnino…). Dopodichè viene raggiunta una pagina nella quale inserire i dati da modificare (mediante apposite form) e che esegue la modifica vera e propria mediante query da mandare al database (sempre con l'apposito comando) . Se ci fossero degli errori durante l'esecuzione della query, questi verrebbero mostrati in output.
- CANCELLAZIONE: Ad ogni cancellazione corrisponde una pagina nella quale scegliere chi/cosa cancellare (Personale, Clienti, Attività) dalla base di dati. Anche qui se ci fossero degli errori durante l'esecuzione della query, questi verrebbero mostrati in output.
- VISUALIZZAZIONE: Per ogni tabella della base di dati corrisponde una pagina php che ha il compito di interrogare la base di dati mediante una query che restituisce tutti i valori della tabella interessata (con qualche lieve modifica per rendere il tutto più 'leggibile').