**ESEMPIO DI SOLUZIONE ESAME DI STATO 2013/2014**
**INFORMATICA**
**(Testo valevole per i corsi di ordinamento e per i corsi sperimentali del Progetto "Sirio")**



La traccia di esame, pubblicata dal Ministero, si trova al seguente link: esame 2014

# 1 Analisi

*Un'analisi della realtà di riferimento individuando le possibili soluzioni e scelga quella che a suo motivato giudizio è la più idonea a rispondere alle specifiche indicate*

Quest'anno, viene illustrato un virtuoso sistema di gestione dei percorsi di alternanza Scuola-Lavoro destinato proprio agli studenti degli Istituti Tecnici.

La traccia, chiara e ben scritta, descrive un sistema gestito e residente all'interno di una istituzione scolastica. Il sistema prevede come attori principali l'azienda e gli studenti, su questo punto credo che pochi studenti abbiano avuto dei dubbi.

Le *aziende* dovranno prevedere dei *periodi*, data inizio e data fine, in cui gli *studenti* lavoreranno presso di loro coordinati da un *tutor*. Non ci sono possibilità di fraintendimenti: il tutor è un componente dell'azienda e non deve essere confuso con i docenti della scuola.
Alla fine del lavoro presso l'azienda, deve venire rilasciato un attestato che riporterà le attività volte e sia i nominativi delle attività svolte che del tutor e del docente referente.

Ogni candidato, in questo primo punto di analisi, ha la possibilità di giustificare le decisioni che prenderà in seguito sul disegno della base dati. Per i più preparati potrebbe essere un'occasione per mostrare una capacità di approfondimento e di vaglio di varie soluzioni che potrebbe comportare una maggiore valutazione dell'elaborato.

L'analisi è un punto di difficile realizzazione e, nella pratica, si traduce spesso con un riassunto della traccia.
Nella realizzazione del modello concettuale riporterò almeno tre diverse possibili soluzioni che potevano anche essere analizzate in questa prima fase.

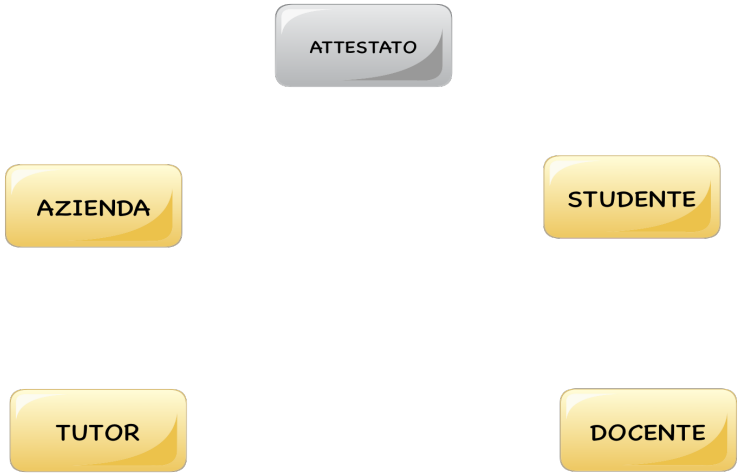
# 2 Modello concettuale

*uno schema concettuale della base di dati;*

Il cuore della progettazione di una base dati. Nel momento in cui si è sviluppato un modello concettuale coerente e completo, il resto della traccia viene di conseguenza a cascata.

Da una prima lettura della traccia due entità cardine appaiono chiare:


Rileggendo la traccia si iniziano a delineare le altre entità:



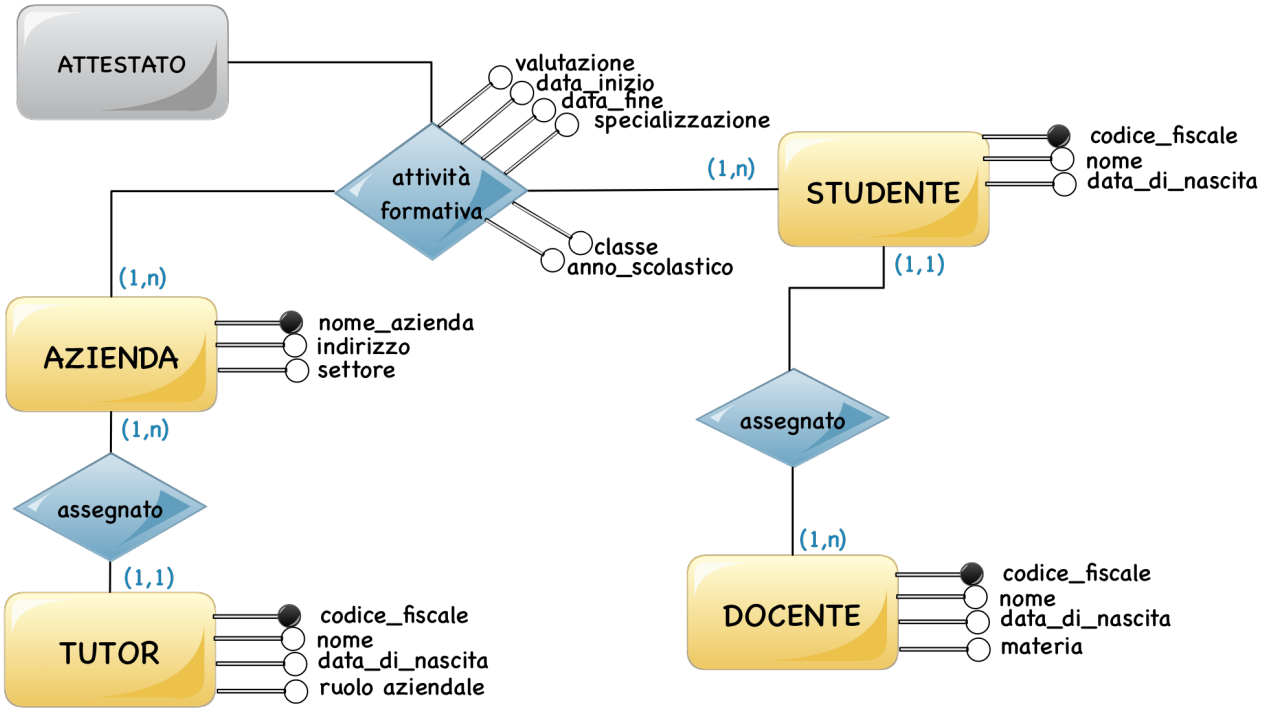

Resta il dubbio sull'attestato, colorata in grigio. Ragionandoci probabilmente risulta ridondante e, alla fine, anche inutile.

Leggendo le query sorgono dei dubbi. Davvero queste quattro entità sono sufficienti? Si parla, nelle due query *c* e *d*, esplicitamente dell'*anno scolastico* e della classe che i ragazzi frequentano. Per indurre a ragionare il candidato, vengono espressamente nominati due anni scolastici differenti (2012/2013 e 2013/2014). Se mettessimo sull'entità *studente*, come verrebbe naturale, sia la *classe* di appartenenza che l'*anno scolastico*, avremo perso la storia di quello che lo stesso studente ha realizzato l'anno prima, perché in maniera del tutto naturale i ragazzi ogni anno cambiano classe e sarebbe sbagliato ripetere e duplicare i nomi degli studenti.

A cosa serve indicare l'a*nno scolastico* e la *classe* dello studente? Rileggendo le query non ci sono dubbi, sono legate alle *attività formative* presso l'*azienda* e, riflettendoci, probabilmente non interesserebbe la 'sezione' della classse ma il livello di avanzamento di studio (terza, quarta e quinta). Non essendoci in merito delle specifiche, personalmente la decisione più coerente, e che semplifica lo svolgimento nelle query, è inserire la '*classe*' sull'*attività formativa* e per l'*anno scolastico* si può decidere: o si risolve con un *between* tra le date (settembre 2012, giugno 2013) o, per semplicità, visto che una query richiederà una *group by* proprio sull'*anno scolastico*, si aggiunge come attributo, sempre sulle *attività formative*.

Un modello concettuale completo dovrebbe essere come quello seguente:

**Modello concettuale numero 1**

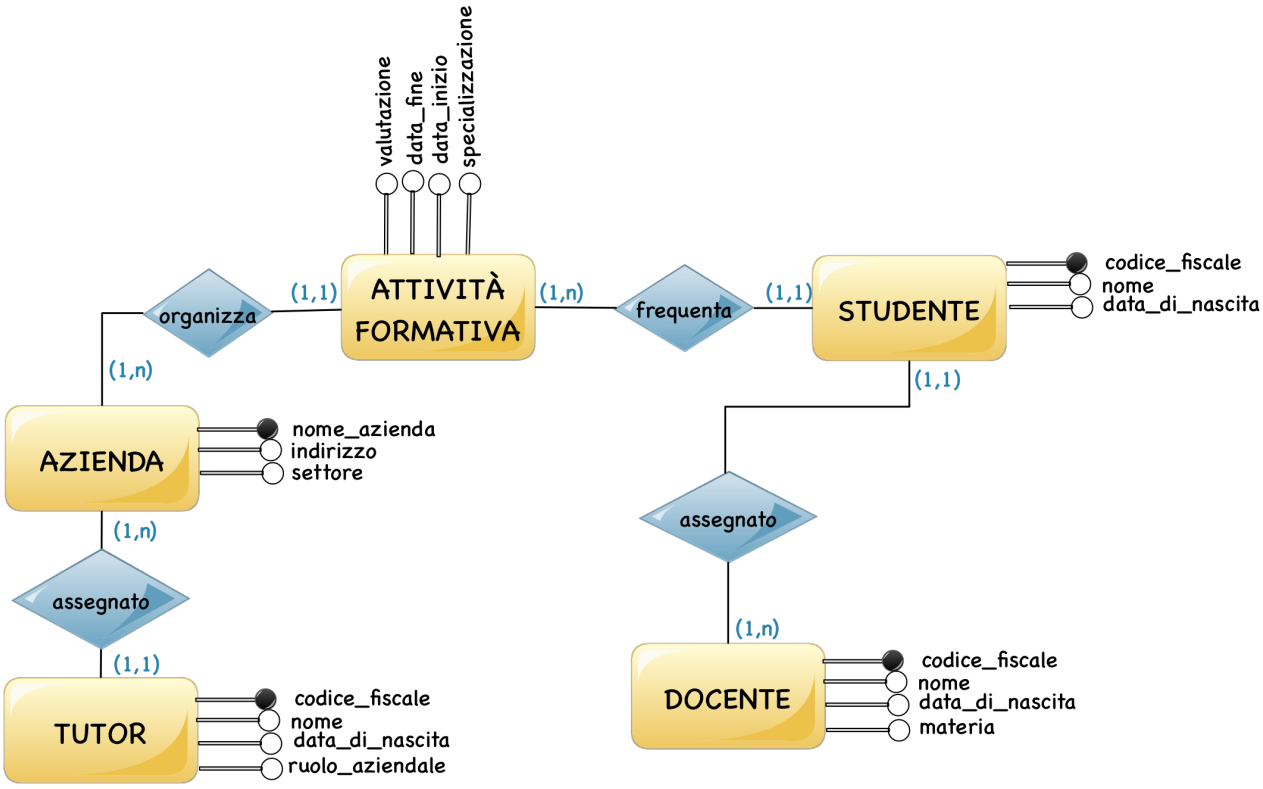



L'attestato non diventerà un'entità. Probabilmente sarà una vista, ma come entità sarebbe ridondante.

Personalmente, non lo considero un errore inserire, al posto della relazione che lega studente e azienda, una ulteriore entità ATTIVITA_FORMATIVA oppure una entità CLASSE.

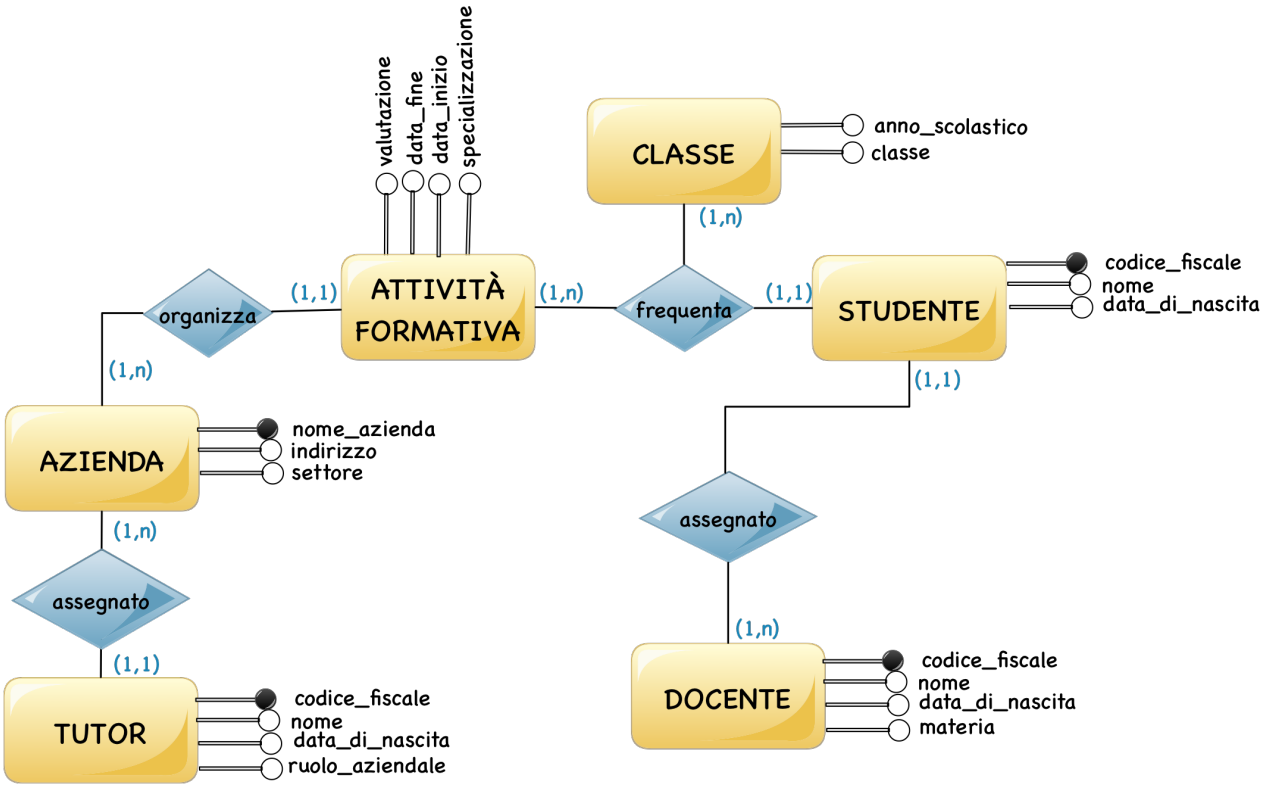
In questo caso il diagramma diventerebbe come la figura seguente:

**Modello concettuale numero 2**



O nel caso si volesse aggiungere l'entità CLASSE:

**Modello concettuale numero 3**




Attenzione: Se si è scelta questa soluzione, non si deve cadere nella trappola di definire tra STUDENTE e ATTIVITA FORMATIVA una relazione molti-a-molti. Si deve considerare che uno studente, appartenente ad una sola classe, può seguire un corso alla volta all'interno di quella determinata azienda per un certo periodo e quindi si deve definire bene la primary key. Nel caso in cui si considerasse che lo studente può frequentare **n** corsi, nel modello logico ci troveremo una entità in più che non ha motivo di esistere. Il corso è univoco per uno studente, un'azienda e il suo determinato periodo di svolgimento.

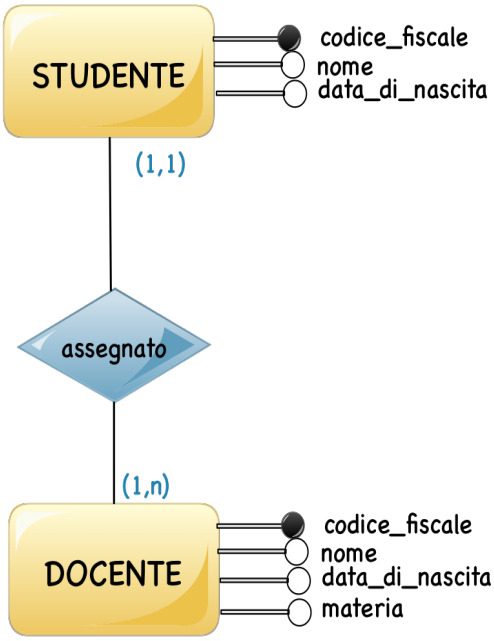
Porteremo avanti il **modello concettuale numero 1**, che preferisco per compattezza, semplicità e soprattutto rende lo svolgimento delle query molto agevole.
Spesso, nella progettazione reale della base dati, si tende a non normalizzare all'eccesso per avere una maggiore performance sull'esecuzione delle query.

## *2.1 Le relazioni*

Vale spendere qualche riga sulle motivazioni che giustificano le cardinalità delle relazioni, su cui molti ragazzi hanno avuto dei dubbi.
Si fa riferimento al **modello concettuale numero 1**.

### 2.1.1 Relazione Studente Docente

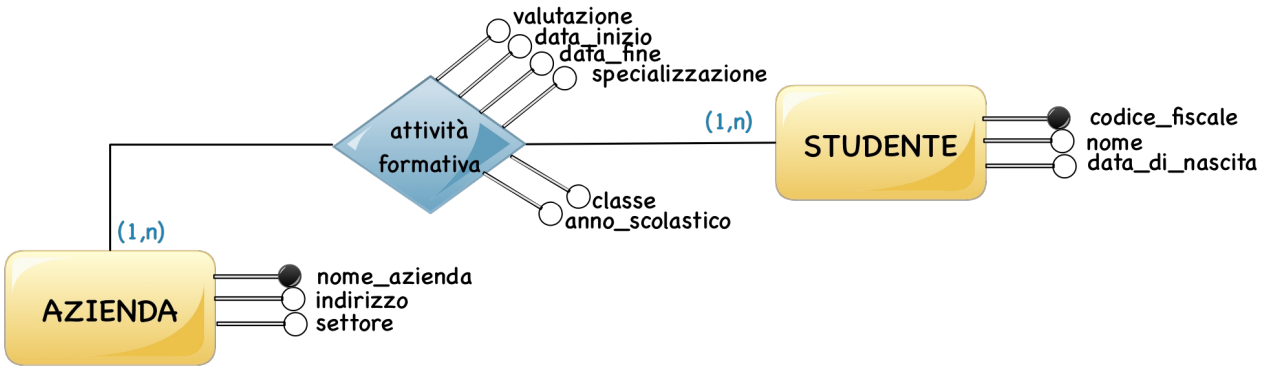
Lo *studente* e il *docente* sono legati tra loro da una relazione *uno a molti*. Un docente fa da referente per più studenti.




### 2.1.2 Relazione Azienda e Studente

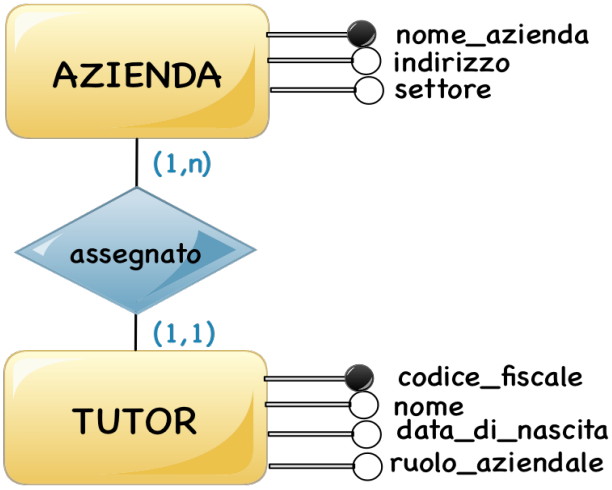
Non ci sono dubbi sul fatto che all'interno di un'azienda ci siano **n** studenti.
Resta un dubbio se uno studente possa frequentare più aziende.

La frase: «*Gli studenti possono essere inviati a più riprese presso le aziende effettuando, in tal modo, più di un periodo di formazione*» fa propendere per l'ipotesi che gli studenti possano essere affidati a diverse aziende e non sempre alla stessa. Per questo motivo si considera, nel caso del primo modello concettuale, una relazione **molti a molti**.




### 2.1.3 Relazione tra Azienda e Tutor

Di sicuro il tutor appartiene ad una sola azienda. D'altro canto rimane un dubbio: c'è un solo tutor per azienda? La risposta è nel buon senso. Sicuramente una grande azienda nominerà più di un tutor per seguire gli studenti. La relazione tra azienda e tutor quindi è di **uno a molti**.




## 2.2 Attributi

Gli attributi delle entità sono abbastanza chiari all'interno della prima parte della traccia. Si devono aggiungere, leggendo il testo delle query la '**specializzazione**', mai nominata nella traccia, che probabilmente riguarda le mansioni lavorative dello studente all'interno dell'azienda. Per questo motivo dovrebbe esserci un attributo inserito sull'entità '*attività lavorativa*' e su questo non ci devono essere fraintendimenti. Lo *studente*, facendo vari periodi, potrebbe anche acquisire *specializzazioni* diverse.

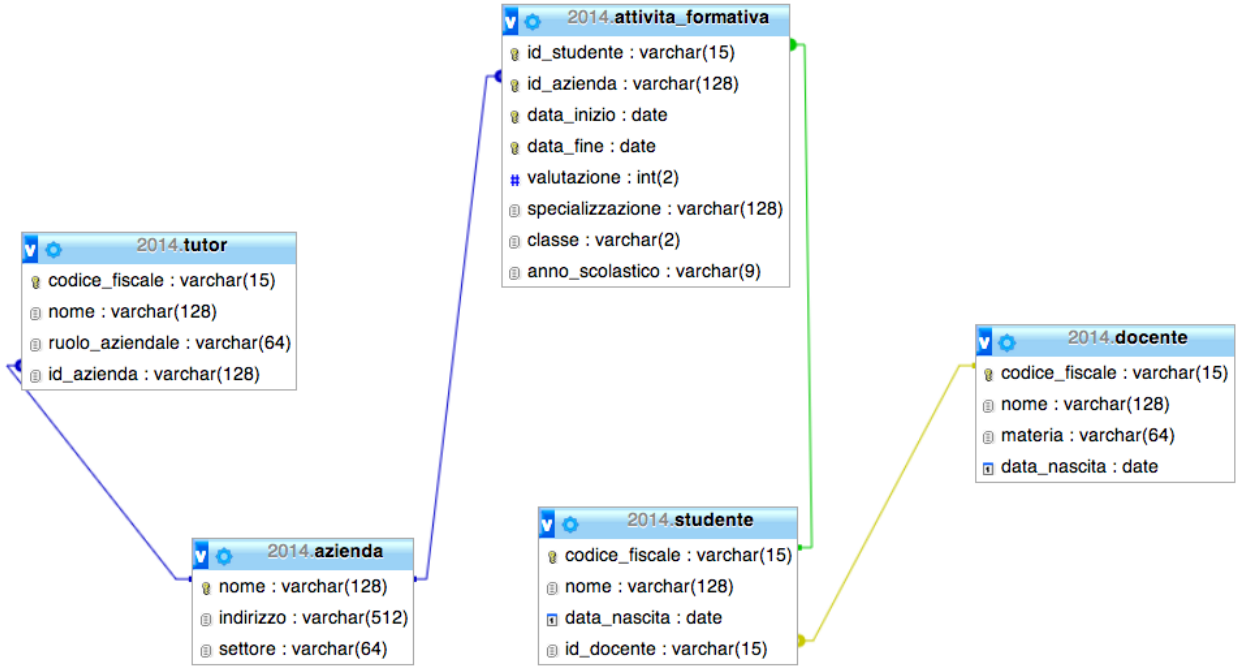
Dalle query si evince anche la necessità dell'attributo '**anno scolastico**'.

Probabilmente, molti avranno optato per la semplice soluzione di inserire l'*anno scolastico* nell'entità *studente*, collegandolo insieme alla *classe*. Riportiamo al ragionamento fatto prima sulla considerazione che classe e anno scolastico non possono essere legati allo studente, che ogni anno cambia classe.

Come precedente detto, non sarebbe un errore considerare una entità *classe*, con attributi classe, *sezione* e *anno scolastico*, giustificata dalla considerazione che gli studenti ogni anno cambiano classe e che le operazioni di assegnazione studente-classe sarebbero state più semplici e la base dati più normalizzata. Risulta, d'altro canto, più efficiente avere questi attributi legati all'*attività lavorativa*.

# 3 MODELLO LOGICO

Creato il modello concettuale, sempre riferito al numero 1, il modello logico ne consegue.




Vengono aggiunte le foreign key, le primary key e viene aggiunta la tabella *attività_formativa* che conterrà le foreign key sia a studente che ad azienda.

La relazione uno a uno tra tutor e azienda si scioglie con la considerazione che l'azienda esiste anche senza nominare alcun tutor, mentre un tutor ha senso di esistere solo nominato dalla sua azienda, per cui sarà il tutor a contenere la foreign key verso la sua azienda.

**azienda** (nome, indirizzo, settore) ;

**docente** (codice_fiscale, nome, materia, data_nascita) ;

**studente** (codice_fiscale, nome, data_nascita date, id_docente) ;
FOREIGN KEY ( id_docente) REFERENCES docente(codice_fiscale)

**attivita_formativa** (id_studente, id_azienda, data_inizio, data_fine, valutazione, specializzazione, classe, anno_scolastico);
FOREIGN KEY (id_studente) REFERENCES studente(codice_fiscale),
FOREIGN KEY (id_azienda) REFERENCES azienda(nome)

**tutor** (codice_fiscale,nome,ruolo_aziendale,id_azienda);
FOREIGN KEY (id_azienda) REFERENCES azienda(nome)

# 4 la definizione delle relazioni della base di dati in linguaggio SQL

Di seguito gli script SQL che sintetizzano le relazioni tra le entità SQL:

```
CREATE TABLE IF NOT EXISTS `azienda` (
  `nome` varchar(128) NOT NULL,
  `indirizzo` varchar(512) NOT NULL,
  `settore` varchar(64) NOT NULL,
  PRIMARY KEY (`nome`)
) ;

CREATE TABLE IF NOT EXISTS `docente` (
  `codice_fiscale` varchar(15) NOT NULL,
  `nome` varchar(128) NOT NULL,
  `materia` varchar(64) NOT NULL,
  `data_nascita` date NOT NULL,
  PRIMARY KEY (`codice_fiscale`)
) ;

CREATE TABLE IF NOT EXISTS `studente` (
  `codice_fiscale` varchar(15) NOT NULL,
  `nome` varchar(128) NOT NULL,
  `data_nascita` date NOT NULL,
  `id_docente` varchar(15) NOT NULL,
   PRIMARY KEY (`codice_fiscale`),
   FOREIGN KEY (`id_docente`)  REFERENCES docente(`codice_fiscale`)
) ;

CREATE TABLE IF NOT EXISTS `attivita_formativa` (
  `id_studente` varchar(15) NOT NULL,
  `id_azienda` varchar(128) NOT NULL,
  `data_inizio` date NOT NULL,
  `data_fine` date NOT NULL,
  `valutazione` int(2) NOT NULL,
  `specializzazione` varchar(128) NOT NULL,
  `classe` varchar(2) DEFAULT NULL,
  `anno_scolastico` varchar(9) DEFAULT NULL,
   PRIMARY KEY (`id_studente`,`id_azienda`,`data_inizio`,`data_fine`),
   FOREIGN KEY (`id_studente`)  REFERENCES  studente(`codice_fiscale`),
   FOREIGN KEY (`id_azienda`)  REFERENCES  azienda(`nome`)
);

CREATE TABLE IF NOT EXISTS `tutor` (
  `codice_fiscale` varchar(15) NOT NULL,
  `nome` varchar(128) NOT NULL,
  `ruolo_aziendale` int(11) NOT NULL,
  `id_azienda` varchar(128) NOT NULL,
  PRIMARY KEY (`codice_fiscale`),
  FOREIGN KEY (`id_azienda`)  REFERENCES  azienda(`nome`)
  );
```

# 5 Le query SQL

a. *elencare le aziende ed i relativi tutor*

La prima query è la più semplice e credo che nessuno studente abbia avuto difficoltà nella sua formulazione. Si tratta di una semplice join tra due tabelle unite dalle loro chiavi esterne senza ulteriori condizioni di selezione:

```
SELECT * FROM azienda, tutor WHERE azienda.nome = tutor.id_azienda ORDER BY id_azienda;
```

```
SELECT *
FROM azienda, tutor
WHERE azienda.nome = tutor.id_azienda
LIMIT 0 , 30
```

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

+ Options

| nome | indirizzo | settore | codice_fiscale | nome | ruolo_aziendale | id_azienda |
|---|---|---|---|---|---|---|
| Azienda 1 | Via delle aziende n. 1 | Informatico | TUTCCC12345631 | Tutor 1 | Responsabile Risorse Umane | Azienda 1 |
| Azienda 2 | Via delle aziende n. 2 | Informatico | TUTCCC12345632 | Tutor 2 | Responsabile Risorse Umane | Azienda 2 |
| Azienda 3 | Via delle aziende n. 3 | Informatico | TUTCCC12345633 | Tutor 3 | Responsabile Risorse Umane | Azienda 3 |
| Azienda 4 | Via delle aziende n. 4 | Informatico | TUTCCC12345634 | Tutor 4 | Responsabile Risorse Umane | Azienda 4 |
| Azienda 5 | Via delle aziende n. 5 | Informatico | TUTCCC12345635 | Tutor 5 | Responsabile Risorse Umane | Azienda 5 |
| Azienda 6 | Via delle aziende n. 6 | Informatico | TUTCCC12345636 | Tutor 6 | Responsabile Risorse Umane | Azienda 6 |
| Azienda 7 | Via delle aziende n. 7 | Informatico | TUTCCC12345637 | Tutor 7 | Responsabile Risorse Umane | Azienda 7 |
| Azienda 8 | Via delle aziende n. 8 | Informatico | TUTCCC12345638 | Tutor 8 | Responsabile Risorse Umane | Azienda 8 |
| Azienda 9 | Via delle aziende n. 9 | Informatico | TUTCCC12345639 | Tutor 9 | Responsabile Risorse Umane | Azienda 9 |

b. *elencare gli studenti delle classi quinte che partecipano all'attività di alternanza, suddivisi per specializzazione*

La seconda query richiedeva di valutare questo attributo *specializzazione* che ha mandato in crisi molti candidati.

Personalmente avevo interpretato la *specializzazione* come la mansione svolta presso l'azienda. Inserita come attributo nell'entità *attività_formativa* diveniva anche questa una query semplice.

Ho avuto assicurazione che, invece, la *specializzazione* si riferisce all'indirizzo di studio (meccanici, elettronici, informatici, ecc..). Avendo inserito *classe* e *anno scolastico*, inseriremo questo ulteriore campo su attività formativa. In questo caso sono tre attributi che in effetti sono attributi della classe di appartenenza degli studenti e non dell'attività presso l'azienda. Appare evidente che la base dati presenta molti dati ripetuti, non è normalizzata e probabilmente l'entità *classe* sarebbe necessaria (modello concettuale 3).

L'unico neo della query è che non viene specificato l'anno scolastico, forse si intende l'anno corrente.

Vengono svolte le due interpretazioni. Per tutti gli anni scolastici:

```
SELECT specializzazione, nome, classe, anno_scolastico
FROM studente, attivita_formativa
WHERE studente.codice_fiscale = attivita_formativa.id_studente
      AND classe LIKE '5%' ORDER BY specializzazione;
```

```
SELECT specializzazione, nome, classe, anno_scolastico
FROM studente, attivita_formativa
WHERE studente.codice_fiscale = attivita_formativa.id_studente
AND classe LIKE '5%'
ORDER BY specializzazione
LIMIT 0 , 30
```

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

+ Options

| specializzazione | nome | classe | anno_scolastico |
|---|---|---|---|
| Programmatore Java | Studente12 | 5H | 2012/2013 |
| Programmatore Java | Studente13 | 5C | 2013/2014 |
| Programmatore Java | Studente18 | 5G | 2013/2014 |
| Programmatore PHP | Studente 1 | 5G | 2012/2013 |
| Programmatore PHP | Studente 1 | 5G | 2013/2014 |
| Programmatore Web | Studente 1 | 5G | 2012/2013 |
| Programmatore Web | Studente 1 | 5G | 2013/2014 |
| Web Designer | Studente12 | 5H | 2012/2013 |

Per l'anno scolastico corrente:

```
SELECT specializzazione,nome, classe
FROM studente, attivita_formativa
WHERE studente.codice_fiscale = attivita_formativa.id_studente
AND classe LIKE '5%' AND anno_scolastico='2013/2014' ORDER BY specializzazione;
```

```
SELECT specializzazione, nome, classe
FROM studente, attivita_formativa
WHERE studente.codice_fiscale = attivita_formativa.id_studente
AND classe LIKE '5%'
AND anno_scolastico = '2013/2014'
ORDER BY specializzazione
LIMIT 0 , 30
```

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

+ Options

| specializzazione | nome | classe |
|---|---|---|
| Programmatore Java | Studente13 | 5C |
| Programmatore Java | Studente18 | 5G |
| Programmatore PHP | Studente 1 | 5G |
| Programmatore Web | Studente 1 | 5G |

**c.** *determinare il numero di studenti che ciascuna azienda ha accolto nell'anno scolastico 2013/2014;*

Viene richiesto il numero di studenti, per cui si deve pensare ad una SELECT COUNT() sugli studenti raggruppati per azienda.

Per il disegno che personalmente ho considerato, la query è relativamente semplice anche in considerazione del fatto che la primary key dell'azienda è il nome stesso e non un id, per cui non c'è la necessità di andare in join sulla tabella azienda.

```
SELECT id_azienda, COUNT( studente.codice_fiscale ) AS numero_studenti
FROM studente, attivita_formativa
WHERE studente.codice_fiscale = attivita_formativa.id_studente
AND anno_scolastico =  '2013/2014'
GROUP BY id_azienda
```

```
SELECT id_azienda, COUNT( studente.codice_fiscale ) AS numero_studenti
FROM studente, attivita_formativa
WHERE studente.codice_fiscale = attivita_formativa.id_studente
AND anno_scolastico = '2013/2014'
GROUP BY id_azienda
LIMIT 0 , 30
```

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

+ Options

| id_azienda | numero_studenti |
|---|---|
| Azienda 1 | 2 |
| Azienda 2 | 3 |
| Azienda 3 | 1 |
| Azienda 4 | 1 |

**d.** *stabilire la classe con il maggior numero di studenti che, nell'anno scolastico 2012/2013, hanno frequentato i percorsi di alternanza*

Partendo dalla query precedente, si imposta la seguente query e aggiungendo il raggruppamento per anno scolastico e classe. Di tutti gli studenti e le classi, si seleziona il max e la relativa classe.

```
SELECT MAX( numero ) , class
FROM (
SELECT COUNT( studente.codice_fiscale ) AS numero, classe AS class
FROM studente, attivita_formativa
WHERE studente.codice_fiscale = attivita_formativa.id_studente
GROUP BY anno_scolastico, classe
HAVING anno_scolastico =  '2012/2013'
) AS numeri
```

```
SELECT MAX( numero ) , class
FROM (

  SELECT COUNT( studente.codice_fiscale ) AS numero, classe AS class
  FROM studente, attivita_formativa
  WHERE studente.codice_fiscale = attivita_formativa.id_studente
  GROUP BY anno_scolastico, classe
  HAVING anno_scolastico = '2012/2013'
```

Show : Start row: 0 Number of rows: 30 Headers every 100

+ Options

| MAX( numero ) | class |
|---|---|
| 5 | 5G |

**e.** *stampare gli attestati relativi ai periodi di formazione a cui ha partecipato un singolo studente;*

Query semplice a cui ho aggiunto delle stringhe di testo fisso visto che non viene chiesto l'elenco ma la 'stampa' dell'attestato.

```
SELECT "Per il periodo dal:", data_inizio,
" al:, data_fine,
" lo studente: ", nome,
" ha acquisito la specializzazione in: ",specializzazione,
" presso: ", id_azienda,
" con valutazione finale: ", valutazione
FROM studente, attivita_formativa WHERE studente.codice_fiscale =
attivita_formativa.id_studente
```

```
SELECT "Per il periodo dal:", data_inizio, " al:", data_fine, " lo studente: ", nome, " ha acquisito la specializzazione in: ", specializzazione, " presso: ", id_azienda,
" con valutazione finale: ", valutazione
FROM studente, attivita_formativa
WHERE studente.codice_fiscale = attivita_formativa.id_studente
LIMIT 0 , 30
```

Profiling [Inline] [ Edit ] [ Explain SQL ] [ Create PHP Code ] [ Refresh ]

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

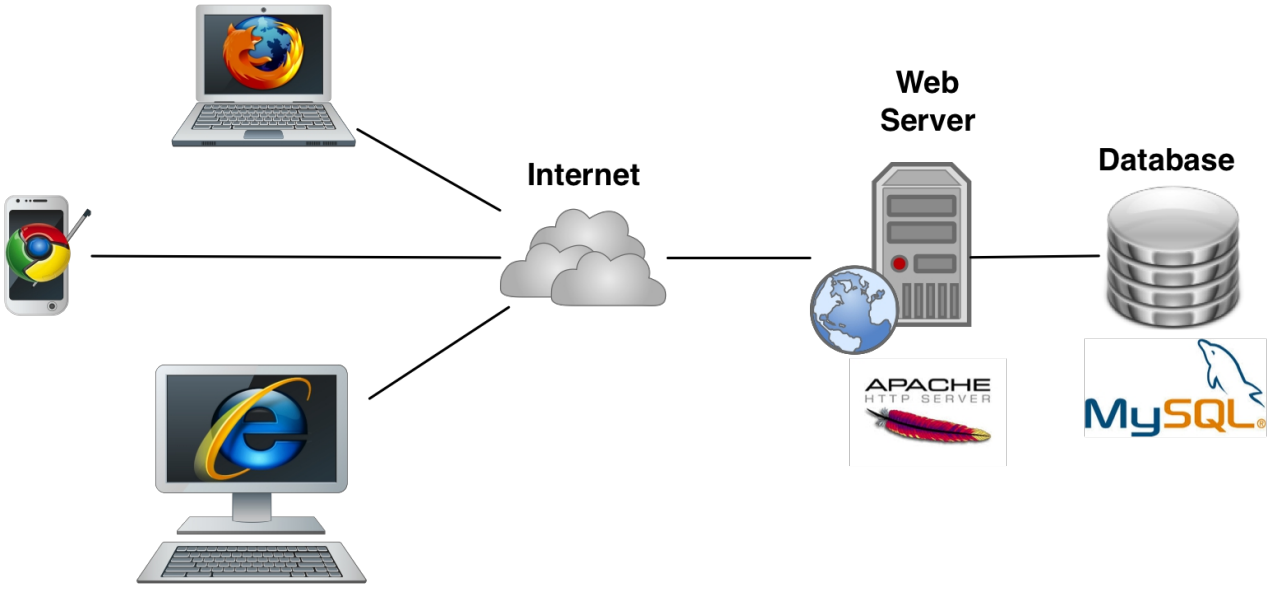
+ Options

| Per il periodo dal: | data_inizio | al: | data_fine | lo studente: | nome | ha acquisito la specializzazione in: | specializzazione | presso: | id_azienda | con valutazione finale: | valutazione |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Per il periodo dal: | 2014-03-01 | al: | 2014-03-30 | lo studente: | Studente 1 | ha acquisito la specializzazione in: | Programmatore PHP | presso: | Azienda 3 | con valutazione finale: | 10 |
| Per il periodo dal: | 2014-05-01 | al: | 2014-05-30 | lo studente: | Studente 1 | ha acquisito la specializzazione in: | Programmatore PHP | presso: | Azienda 3 | con valutazione finale: | 10 |
| Per il periodo dal: | 2013-10-01 | al: | 2013-10-30 | lo studente: | Studente 1 | ha acquisito la specializzazione in: | Programmatore Web | presso: | Azienda 4 | con valutazione finale: | 8 |
| Per il periodo dal: | 2013-12-01 | al: | 2013-12-30 | lo studente: | Studente 1 | ha acquisito la specializzazione in: | Programmatore Web | presso: | Azienda 4 | con valutazione finale: | 8 |
| Per il periodo dal: | 2013-10-01 | al: | 2013-10-30 | lo studente: | Studente11 | ha acquisito la specializzazione in: | Programmatore Java | presso: | Azienda 1 | con valutazione finale: | 10 |
| Per il periodo dal: | 2013-03-01 | al: | 2013-03-30 | lo studente: | Studente11 | ha acquisito la specializzazione in: | Programmatore PHP | presso: | Azienda 3 | con valutazione finale: | 10 |
| Per il periodo dal: | 2012-10-01 | al: | 2012-10-30 | lo studente: | Studente11 | ha acquisito la specializzazione in: | Programmatore Web | presso: | Azienda 4 | con valutazione finale: | 8 |
| Per il periodo dal: | 2013-10-01 | al: | 2013-10-30 | lo studente: | Studente12 | ha acquisito la specializzazione in: | Programmatore Java | presso: | Azienda 1 | con valutazione finale: | 10 |
| Per il periodo dal: | 2012-10-01 | al: | 2012-10-30 | lo studente: | Studente12 | ha acquisito la specializzazione in: | Web Designer | presso: | Azienda 5 | con valutazione finale: | 9 |
| Per il periodo dal: | 2013-10-01 | al: | 2013-10-30 | lo studente: | Studente12 | ha acquisito la specializzazione in: | Web Designer | presso: | Azienda 5 | con valutazione finale: | 9 |
| Per il periodo dal: | 2012-10-01 | al: | 2012-10-30 | lo studente: | Studente13 | ha acquisito la specializzazione in: | Programmatore C++ | presso: | Azienda 1 | con valutazione finale: | 10 |
| Per il periodo dal: | 2013-10-01 | al: | 2013-10-30 | lo studente: | Studente13 | ha acquisito la specializzazione in: | Programmatore Java | presso: | Azienda 1 | con valutazione finale: | 10 |
| Per il periodo dal: | 2012-10-01 | al: | 2012-10-30 | lo studente: | Studente17 | ha acquisito la specializzazione in: | Programmatore Java | presso: | Azienda 2 | con valutazione finale: | 10 |
| Per il periodo dal: | 2013-10-01 | al: | 2013-10-30 | lo studente: | Studente17 | ha acquisito la specializzazione in: | Programmatore Java | presso: | Azienda 2 | con valutazione finale: | 10 |
| Per il periodo dal: | 2013-10-01 | al: | 2013-10-30 | lo studente: | Studente18 | ha acquisito la specializzazione in: | Programmatore Java | presso: | Azienda 2 | con valutazione finale: | 10 |
| Per il periodo dal: | 2013-10-01 | al: | 2013-10-30 | lo studente: | Studente19 | ha acquisito la specializzazione in: | Programmatore Java | presso: | Azienda 2 | con valutazione finale: | 10 |

# 6 L’interfaccia utente e un pezzo di codice

Il Sistema informativo proposto potrebbe essere realizzato come applicazione web, in modo da poter essere utilizzato sia all’interno della scuola, sia dalle aziende che nominano i tutor e valutano gli alunni che lavorano presso di loro.

Realizzare un'applicazione Web si traduce nel mettere su di un WEB SERVER delle pagine fruibili da un browser (siano esse html, php, asp, jsp etc.)
L’ architettura di riferimento più economica, solida, che risponde agli standard di mercato ed al limitato numero di utenti da gestire, è quella disegnata nella figura che segue: APACHE, linguaggio PHP e Base Dati MySQL.




L'interfaccia sarà realizzata in HTML5 con l'ausilio del PHP per quanto riguarda le interazioni con il database.
Nel momento in cui si realizza un'applicazione web, all'interno della base dati si devono prevedere delle tabelle di profilazione degli utenti che possono agire sul sistema.
Gli utenti dovranno registrarsi con *username* ed una *password* e avranno vari livelli di autorizzazioni. I docenti potranno inserire e modificare i dati delle attività formativa, gli utenti autorizzati delle aziende potranno nominare i loro tutor ed inserire specializzazioni e valutazione degli utenti, mentre gli studenti potranno solo visualizzare i dati che riguardano le loro attività.
Questo tipo di base dati esula dalla traccia, per cui proponiamo l'interfaccia della richiesta di un attestato e la sua visualizzazione sfruttando una query già richiesta dalla traccia.
L'importante di questi punti molto tecnici, che solo nei compiti di informatica scritti si eseguono senza l'ausilio di un PC, personalmente credo sia l'impostazione corretta. Ovvero sapere di massima com'è strutturata una pagina HTML e come si programma un modulo PHP.

## 6.1 Richiedi attestato per studente e periodo

Presupponiamo che l'utente, che sta richiedendo un attestato, si sia precedentemente autenticato e sia o lo studente stesso o il docente referente.

```
<html>
<head>
<title>Richiedi Attestato</title>
</head>
<body>
<p> <b>Richiedi Attestato</b></p>
<form action="attestato.php" method="POST" name="form">
<table width="160" border="0">

  <tr>
    <td><label>Nome Studente</label> </td>
    <td><input name="nome" id="nome" type="text" size="10" maxlength="10"
/> </td>
  </tr>
  <tr>
    <td><label>Periodo dal</label> </td>
    <td><input name="dal" id="dal" type="text" size="10" maxlength="10"
/> </td>
  </tr>
  <tr>
```

```
    <td><label>al</label> </td>
    <td><input name="al" id="al" type="text" size="10" maxlength="10"
/> </td>
  </tr>
 <tr >
    <td colspan="2" align="center"><input name="submit" type="submit"
value="INVIO" /> <input name="reset" type="reset" value="RESET" /></td>
 </tr>
</table>

</form>

</body>
</html>
```

## 6.2 File attestato.php

```
      <html>
         <head>
                  Stampa attestato <br>
            </head>
            <body>
                  <?php

      $nomestudente=$_POST[nome];
      $dal=$_POST[dal];
      $al=$_POST[dal];

      $db=mysql_connect("localhost","root","root");

      if(!$db)
            echo "no connection";
      else
      {
            mysql_select_db("my_giselda");
            $query = "SELECT
            nome,
            specializzazione,
             id_azienda,
             valutazione
            FROM studente, attivita_formativa WHERE studente.codice_fiscale =
                        attivita_formativa.id_studente
            and nome = '%".$nome."%' and data_inizio = '".$dal."' and data_fine
= '".$al."';"


            $result = mysql_query($query);
            $num_results = mysql_num_rows($result);



            for ($i=0; $i < $num_results; $i++)
            {
                  $row = mysql_fetch_array($result);
                  echo "<br>Nome Studente: ".$row[nome]." <br> Periodo dal ".
$dal." al ".$al."<br> Specializzazione in ". $row[specializzazione] ." presso:
". $row[azienda];
            }
      }
            ?>
        </body>
    </html>
```

# 7 L’architettura del sito web

Non ci sono dubbi sull’interpretazione di questo punto. Non viene richiesta l’architettura del sistema informativo, ma l’architettura di un sito Web di ‘presentazione’ del progetto.

Si può immaginare che sul sito di presentazione sia data anche la possibilità, sia alle aziende che agli studenti, di sottoscriversi per ottenere ulteriori informazioni o proprio per dare l'adesione al progetto.

L'architettura è quella vista nel punto precedente, ma un sito studiato per pubblicizzare un certo progetto dovrebbe essere molto più orientato sull'aspetto grafico che sulla progettazione della base dati.

Su questo punto i ragazzi, in teoria, avrebbero potuto scatenare la loro creatività con loghi, layout di pagine web ma, arrivati alla fine del compito, credo che davvero pochi studenti abbiano avuto ancora le energie per pensare a qualche brillante soluzione di marketing per pubblicizzare il sistema.