OBIETTIVO: progettazione di una base dati per una multinazionale farmaceutica che acquista i prodotti da terzi e li immagazzina e rivende presso le varie sedi dislocate in differenti città e nazioni.

Le necessità dell'azienda sono:
- Gestire i dipendenti delle varie sedi, monitorando eventuali scatti di anzianità e ruoli.
- Avere un maggior controllo dei dati dei fornitori e delle relative forniture.
- Controllare gli acquisti e le vendite dei vari prodotti per sede. Questo permetterebbe di ottimizzare le forniture e tenere sotto controllo le giacenze.
- Analizzare i trend dei farmaci (e relative quantità) negli anni/mesi per eventuali analisi predittive.
- Tenere traccia della classificazione dei prodotti e di eventuali nuove classificazioni in base alle normative.

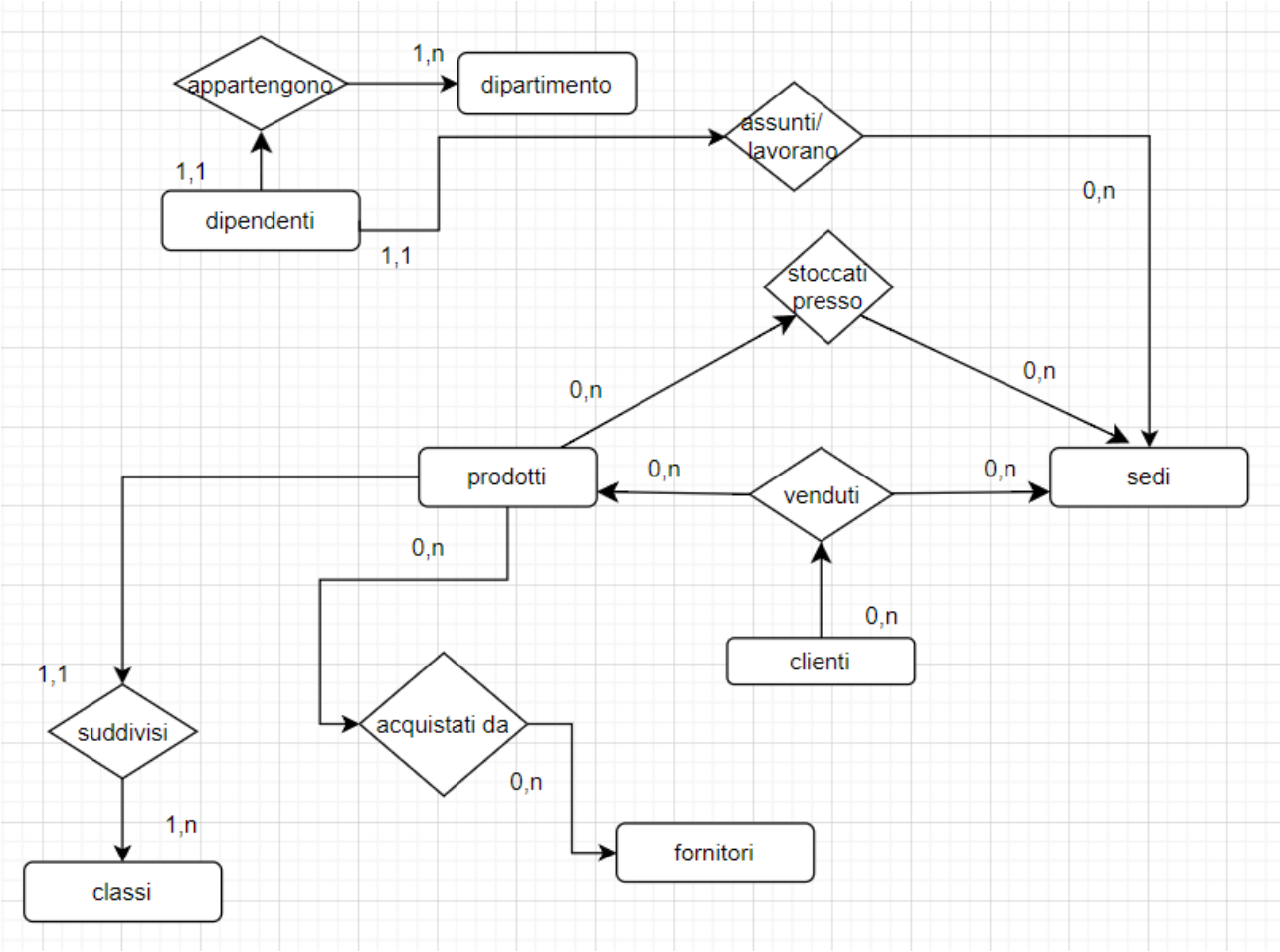

Fig.1 – Modello e molteplicità

In base allo schema in fig.1 ho poi deciso di creare le seguenti tabelle:

DIPARTIMENTI: (**id_dipartimento**, descrizione)

CLASSI: (**id_classe**, descrizione)

FORNITORI: (**id_fornitore**, vat_number, paese, città, indirizzo, telefono, mail)

CLIENTI: (**id_cliente**, n_identificativo, paese, città, indirizzo, telefono, mail)

SEDI: (**id_sede**, vat_number, paese, città, indirizzo, telefono)

DIPENDENTI: (**id_dipendenti**, nome, cognome, ruolo, anno_assunzione, codice_identificativo, paese, città, indirizzo, telefono, mail, salario, *id_sd, id_dipa*)

PRODOTTI: (**id_prodotto**, nome, *id_classe*)

ACQUISTI: (**id_acquisto**, data_acquisto, quantità, prezzo, *id_prod, id_for*)

STOCCAGGI: (**id_stock**, data_in, quantità, *id_prod, id_sd*)

ORDINI: (**id_ordine**, data_ordine, quantità, prezzo, *id_prod, id_cl, id_sd*)

In grassetto sono riportate le **primary keys**, in corsivo-sottolineato le *chiavi esterne* che si riferiscono alle chiavi primarie di altre tabelle.

Le associazioni: “acquistati da” , “stoccati presso” , “venduti” presentano cardinalità N:N; ho trasformato quindi queste relazioni -> in entità con relativi attributi. Per queste entità si potrebbero evitare le chiavi artificiali in quanto potrebbero essere univocamente identificate da data e relative chiavi esterne. Per semplicità, però, ho lasciato nella creazione delle tabelle le chiavi auto incrementanti.

Partecipazione alle relazioni:

- Se un dipendente è stato assunto, deve obbligatoriamente essere associato a un solo dipartimento e una sola sede.
- Se un prodotto è acquistato e venduto dalla multinazionale, deve avere partecipazione obbligatoria nella la relazione con le classi. In pratica, un prodotto deve essere obbligatoriamente classificato.

Semplificazioni che ho utilizzato: i dipendenti possono lavorare in una sola sede.

Vincoli di integrità referenziale: ho scelto per tutte le tabelle il comando “no action” in caso di modifiche o cancellazioni.
In MySql ho creato prima le tabelle senza chiavi esterne e poi quelle che puntavano ad altre tabelle.

CREAZIONE TABELLE

/*database*/

```
create database if not exists multinazionale;

use multinazionale;
```

/* tabelle */

```
create table if not exists dipartimenti (
        id_dipartimento tinyint primary key auto_increment,
        descrizione varchar (150) not null
)
engine = innodb;


create table if not exists fornitori (
      id_fornitore int primary key auto_increment,
      vat_number tinyint not null,
      paese varchar (70) not null,
```

```
        città varchar(80) not null,
        indirizzo varchar (150) not null,
        telefono tinyint not null,
        mail varchar (60) not null
)
engine = innodb;

create table if not exists clienti (
        id_cliente int primary key auto_increment,
        n_identificativo tinyint not null,
        paese varchar (70) not null,
        città varchar(80) not null,
        indirizzo varchar (150) not null,
        telefono tinyint not null,
        mail varchar (60) not null
)
engine = innodb;
create table if not exists classi (
   id_classe tinyint primary key,
   descrizione varchar (150) not null
)
engine = innodb;


create table if not exists prodotti (
   id_prodotto int primary key auto_increment,
   nome varchar(40) not null,
   id_classe tinyint,
   foreign key (id_classe) references classi(id_classe) on update no action on delete no action
)
engine = innodb;


create table if not exists sedi (
   id_sede int primary key auto_increment,
   vat_number tinyint not null,
   paese varchar (70) not null,
   città varchar(80) not null,
   indirizzo varchar (150) not null,
   telefono tinyint,
)
engine = innodb;


create table if not exists dipendenti (
   id_dipendenti int primary key auto_increment,
   nome varchar (60) not null,
   cognome varchar (80) not null,
   anno_assunzione date not null,
   codice_identificativo varchar (30) not null,
   paese varchar (70) not null,
   città varchar(80) not null,
   indirizzo varchar (150) not null,
   telefono tinyint not null,
   mail varchar (60) not null,
```

```
   salario float not null,
   id_sd int not null ,
   id_dipa tinyint ,
   foreign key (id_sd) references sedi(id_sedi) on update no action on delete no action,
   foreign key (id_dipa) references dipartimenti(id_dipartimento) on update no action on delete no action
)
engine = innodb;


create table if not exists acquisti (
   id_acquisto int primary key auto_increment,
   data_acquisto date not null,
   quantita int not null,
   prezzo float not null,
   id_prod int not null,
   id_for int not null,
   foreign key (id_prod) references prodotti(id_prodotto) on update no action on delete no action,
   foreign key (id_for) references fornitori(id_fornitore) on update no action on delete no action
)
engine = innodb;


create table if not exists stoccaggi (
   id_stock int primary key auto_increment,
   data_in date not null,
   quantita int not null,
   id_prod int not null,
   id_sd int not null,
   foreign key (id_prod) references prodotti(id_prodotto) on update no action on delete no action,
   foreign key (id_sd) references sedi(id_sede) on update no action on delete no action
)
engine = innodb;

create table if not exists ordini (
   id_ordine int primary key auto_increment,
   data_ordine date not null,
   quantita int not null,
   prezzo float not null,
   id_cl int not null,
   id_prod int not null,
   id_sd int not null,
   foreign key (id_cl) references clienti(id_cliente) on update no action on delete no action,
   foreign key (id_prod) references prodotti(id_prodotto) on update no action on delete no action,
   foreign key (id_sd) references sedi(id_sede) on update no action on delete no action
)
engine = innodb;
```

/*Due esempi di comandi per l'inserimento dei dati nelle tabelle */

(Avendo delle chiavi auto incrementanti ho specificato sia le colonne che i valori da inserire)

- Per la tabella dipartimenti:
  insert into dipartimenti (descrizione) values ('commerciale');

- Per la tabella stoccaggi:
  insert into stoccaggi (data_in, quantità, id_prod, id_sd) values ('2021-05-13', 200, 4, 8);

VISTE

***/ Visualizzazione date e tot importo degli acquisti fatti per le aspirine*/**

```
CREATE VIEW Acquisti_asprina (data, tot_ordine) as
SELECT acq.data_acquisto, (acq.quantita*acq.prezzo) as tot
FROM acquisti as acq
JOIN prodotti as pd on acq.id_prod = pd.id_prodotto
WHERE pd.nome = 'aspirina';
```

***/ Visualizzazione incassi (senza iva) per sede nel 2023*/**

```
CREATE VIEW Incassi_sedi (sede, tot_ordine) as
SELECT o.id_sd, SUM(o.quantita*o.prezzo) as tot
FROM ordini as o
WHERE YEAR(data_ordine) = ‘2023’
GROUP BY o.id_sd
ORDER BY tot DESC;
```

***/ Visualizzazione n° transazioni inerenti agli acquisti (verso fornitori) divisi per prodotto per il 2023*/**

```
CREATE VIEW Transazioni (id_prodotti, n_transazioni) as
SELECT acq.id_prod, COUNT(id_acquisto) as transazioni
FROM acquisti as acq
WHERE YEAR(data_acquisto) = ‘2023’
GROUP BY acq.id_prod
ORDER BY COUNT(id_acquisto) DESC;
```

QUERY:

1. Dipartimenti con maggiore numero di dipendenti

```
SELECT dipa.id_dipartimento, dipa.descrizione, COUNT(dipe.id_dipendente) as num_dipendenti
FROM dipartimento as dipa JOIN dipendenti as dipe on dipa.id_dipartimento =  dipe.id_dipa
GROUP BY dipa.id_dipartimento
ORDER BY num_dipendenti DESC;
```

2. Sedi con più dipendenti in Italia

```
SELECT sd.id_sede, COUNT(dipe.id_dipendente) as num_dipendenti
FROM sedi as sd JOIN dipendenti as dipe on sd.id_sede =  dipe.id_sd
WHERE sd.paese = "Italia"
GROUP BY sd.id_sede
ORDER BY num_dipendenti DESC;
```

3. Dipendenti con salario maggiore della media del dipartimento amministrativo

p.s la query interna non ha il group by perché non cerco la media per i differenti dipartimenti ma solo il salario medio per quello amministrativo che è già dichiarato nella funzione condizionale WHERE. La query esterna, invece, mi serve per cercare i vari dipendenti che hanno un salario maggiore della media del dipartimento amministrativo (che per es. so essere quello con gli stipendi più alti)

```
SELECT dipe.id_dipendente, dipe.cognome, dipe.ruolo
FROM dipendenti as dipe
WHERE dipe.salario > ( SELECT AVG (dipe.salario),
                       FROM dipendenti as dipe JOIN dipartimento as dipa on dipe.id_dipa = dipa.id_dipartimento
                       WHERE dipa.descrizione = "amministrativo";
```

4. Top 3 prodotti più acquistati (con relativo nome) nell'anno 2021 sulla base dei quantitativi

```
SELECT acq.id_prod, pr.nome, SUM(acq.quantita) as quantita_tot
FROM acquisti as acq JOIN prodotti as pr on acq.id_prod = pr.id_prodotto
WHERE YEAR(acq.data_acquisto) = '2001'
GROUP BY acq.id_prod
ORDER BY quantita_tot DESC
LIMIT 3;
```

5. Nome, cognome, sede dei dipendenti con ruolo 'manager'

```
SELECT dipe.id_dipendente, dipe.nome, dipa.cognome, sd.id_sede
FROM Dipendente as dipe JOIN sedi as sd on dipe.id_sd = sd.id_sede
WHERE dipe.ruolo = 'manager'
ORDER BY sd.id_sede;
```

6. Date in cui sono stati acquistati i prodotti di classe 5

```
SELECT acq.data_acquisto
FROM acquisti as acq JOIN prodotti as pr on acq.id_prod = pr.id_prodotto
WHERE pr.id_classe = 5;
```

7. Anno con maggiori assunzioni

```
SELECT dipe.anno_assunzione, COUNT(id_dipendente) as numero_dipendenti_assunti
FROM dipendenti as dipe
GROUP BY dipe.anno_assunzione
ORDER BY COUNT(id_dipendente) DESC
LIMIT 1;
```

8. Dipartimenti e salario medio per dipartimento per la Germania

```
SELECT dipa.id_dipartimento, AVG(dipe.salario) as salario_medio
FROM dipartimento as dipa JOIN dipendente as dipe on dipa.id_dipartimento = dipe.id_dipa
JOIN sedi as sd on dipe.id_sd = sd.id_sede
WHERE sd.paese = 'Germania'
GROUP BY dipa.id_dipartimento
ORDER BY salario_medio DESC;
```

9. I 3 dipendenti con salario più alto e paese in cui lavorano

```
SELECT dipe.id_dipendente, dipe.salario, sd.paese
```

```
FROM dipendenti as dipe JOIN sedi as sd on dipe.id_sd = sd.id_sede
ORDER BY dipe.salario DESC
LIMIT 3;
```

10. Sedi con stoccaggi in ingresso (quantità) sotto i 50 per i prodotti aventi id = 6 in data 2023-08-22

```
SELECT stock.id_sd as id_sede, SUM(stock.quantita) as quantita_giornaliera_in
FROM stoccaggi as stock
WHERE stock.id_prod = 6 AND data_in = '2023-08-22'
GROUP BY stock.id_sd
HAVING SUM(stock.quantita) < 50;
```

11. Fornitori che hanno fornito il prodotto "ibuprofene" nel 2023

```
SELECT DISTINCT(fr.id_fornitore)
FROM fornitori as fr JOIN acquisti as acq on fr.id_fornitore = acq.id_for
JOIN prodotti as pd on acq.id_prod = pd.id_prodotto
WHERE YEAR(acq.data_acquisto) = '2023' AND pd.nome = 'ibuprofene';
```

12. Prodotti che non sono stati stoccati da più di 60 gg in Italia

```
SELECT stock.id_prod
FROM stoccaggi as stock JOIN sede as sd on stock.id_sd = sd.id_sede
WHERE sd.paese = 'Italia' AND DATEDIFF (NOW(), stock.data_in) > 60 ;
```

13. Dipendenti il cui salario è compreso tra 22.000 e 28.000

```
SELECT dipe.id_dipendenti, dipe.salario
FROM dipendenti as dipe
WHERE dipe.salario BETWEEN 22000 AND 28000
ORDER BY dipe.salario;
```

14. Totale vendite (iva esclusa) nell'anno 2023 divisi per id prodotto

```
 SELECT o.id_prod, sum(o.quantita*o.prezzo) as totale_2023
FROM ordini as o
WHERE year(o.data_ordine) = '2023'
GROUP BY o.id_prod
ORDER BY totale_2023 DESC;
```

15. Top 3 prodotti più venduti nell'anno 2021 sulla base dei quantitativi e relativo numero di transazioni

```
SELECT o.id_prod, sum(o.id_ordine) as tot_quantitativi_venduti, count(o.id_ordine) as n_transazioni
FROM ordini as o
WHERE YEAR(o.data_ordine) = '2001'
GROUP BY o.id_prod
ORDER BY tot_quantitativi_venduti
LIMIT 3;
```