Date le tabelle:

Clienti := < id, nome, cognome, indirizzo,città >
Ordini := < id, data_ora_ordine, id_prodotto, id_cliente, quantità>
Prodotti := < id, nome, descrizione, costo,scorte >

INDICI

Prevediamo di effettuare spesso interrogazioni simili alle seguenti:

Ordini di un certo prodotto effettuati da un cliente:

```
SELECT prodotti.nome,clienti.nome, cognome,quantità,data_ora_ordine
FROM clienti, ordini, prodotti
WHERE   (clienti.id = ordini.id_cliente) AND (ordini.id_prodotto = prodotti.id)
    AND (clienti.nome='Mario' ) AND (clienti.cognome ='Rossi') AND (prodotti.nome='Dixan') ;
```

Ordini effettuati da un cliente:

```
SELECT prodotti.nome,clienti.nome, cognome,quantità,data_ora_ordine
FROM clienti, ordini, prodotti
WHERE   (clienti.id = ordini.id_cliente) AND (ordini.id_prodotto = prodotti.id)
    AND (clienti.nome='Mario' ) AND (clienti.cognome ='Rossi')
```

Ordini di un prodotto effettuati negli ultimi 2 mesi:

```
SELECT prodotti.nome,clienti.nome, cognome,quantità,data_ora_ordine
FROM clienti, ordini, prodotti
WHERE  (clienti.id = ordini.id_cliente) AND (ordini.id_prodotto = prodotti.id)
    AND (prodotti.nome='Dixan') AND (data_ora_ordine> NOW() - INTERVAL 2 MONTH);
```

mentre eseguiremo RARAMENTE cose del tipo:

Ordini di un prodotto effettuati nella città di Milano:

```
SELECT prodotti.nome,clienti.nome, cognome,quantità,data_ora_ordine
FROM clienti, ordini, prodotti
WHERE  (clienti.id = ordini.id_cliente) AND (ordini.id_prodotto = prodotti.id)
    AND (prodotti.nome='Dixan') AND (clienti.città = 'Milano');
```

Conviene creare degli **indici** ovvero delle **strutture dati che permettano di ridurre i tempi necessari per la ricerca**. In particolare visto che effettueremo ricerche su entrambi i campi nome e cognome, creiamo un indice su entrambi i campi (che è diverso dal creare 2 indici su ognuno dei due campi separatamente).
Poiché eseguiremo solo raramente ricerche sul campo città non creiamo un indice su questo campo(**gli indici accelerano la ricerca ma rendono INSERT e UPDATE più lente** in quanto è necessario ricreare o modificare tali strutture ad ogni inserimento o modifica).

Nel caso dei prodotti analogamente possiamo creare un indice sul campo nome.

Mentre nella tabella Ordini dovremo specificare che i campi id_cliente e id_prodotto sono chiavi esterne che fanno riferimento ai campi id delle tabelle clienti e prodotti.

VINCOLI REFERENZIALI :

Specificare che un campo rappresenta una chiave esterna non solo comporta la creazione automatica in MySQL di indici per "velocizzare" le interrogazioni ma ci permette di imporre dei vincoli del tipo:

Se un cliente viene cancellato dalla tabella clienti oppure il valore della chiave primaria viene modificato(es. Se utilizzo come chiave il codice fiscale potrei volerlo modificare se mi accorgo che è sbagliato) che devo fare con la chiave esterna ed i record associati?

Formalmente vi sono essenzialmente 5 tipi di azione possibili:

1) **ON DELETE CASCADE** : se cancello un record "Padre" **vengono cancellati "a cascata" tutti i records "Figli"**. Esempio: se cancello un cliente vengono cancellati anche tutti gli ordini effettuati

2) **ON DELETE RESTRICT** : **impedisco la cancellazione** del record "Padre" **se esso ha dei "Figli"**. Esempio: se un cliente ha effettuato degli ordini, la cancellazione del cliente viene impedita

3) **ON DELETE SET NULL** : se cancello un record "Padre", **la chiave esterna dei records "Figli" associati viene impostata a NULL**(ovviamente se non ho specificato il vincolo che debba essere NOT NULL). Esempio: se cancello un cliente, l' **id_cliente** degli ordini effettuati da quel cliente viene impostato a NULL

4) **ON DELETE NO ACTION**: non viene eseguita alcuna azione

5) **ON DELETE SET DEFAULT** non implementata in MySQL InnoDB

Ovviamente esistono analoghi vincoli nel caso dell' **UPDATE**(cosa faccio se la chiave primaria del record “Padre” viene modificata?).

In conclusione per ogni chiave esterna devono essere specificati 2 vincoli uno sull'UPDATE(cosa fa) e uno sulla DELETE.

Di default MySQL pone **ON DELETE RESTRICT ON UPDATE RESTRICT**

Ma nei casi che vengono incontrati di solito si dovrebbe porre:

ON DELETE RESTRICT ON UPDATE CASCADE

In conclusione ecco le create per le 3 tabelle:

Creazione della tabella **Clienti**:

```
CREATE TABLE  Clienti (
  id int(10) unsigned NOT NULL auto_increment,
  nome varchar(45) NOT NULL,
  cognome varchar(45) NOT NULL,
  indirizzo varchar(45) NOT NULL,
  città varchar(45) NOT NULL,
  PRIMARY KEY  (id),
  INDEX nome_cognome (nome, cognome)    oppure    KEY  nome_cognome (nome, cognome)
)
```

Creazione tabella **Prodotti**

```
CREATE TABLE  Prodotti (
  id int(10) unsigned NOT NULL auto_increment,
  nome varchar(30) NOT NULL,
  descrizione varchar(200) NOT NULL,
  costo decimal(8,2) unsigned NOT NULL,
  scorte int(10) unsigned NOT NULL,
  PRIMARY KEY  (id),
  KEY nome_prod (nome)
);
```

Creazione tabella **Ordini**

```
CREATE TABLE  Ordini (
  id                int(10) unsigned NOT NULL auto_increment,
  id_cliente        int(10) unsigned NOT NULL,
  id_prodotto       int(10) unsigned NOT NULL,
  quantità          int(10) unsigned NOT NULL,
  data_ora_ordine   datetime NOT NULL,

  PRIMARY KEY (id),

  FOREIGN KEY (id_cliente) REFERENCES clienti(id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
  FOREIGN KEY (id_prodotto) REFERENCES prodotti(id)
        ON DELETE RESTRICT ON UPDATE CASCADE
)
```

VINCOLI DI DOMINIO

I vincoli di dominio sono le condizioni che devono essere imposte sul valore degli attributi per salvaguardare l'integrità del database(insieme ai vincoli referenziali e a quelli su gruppi di attributi che vedremo più avanti) ad esempio età NOT NULL, 0< età < 120, ecc... anche in MySQL è possibile specificarli utilizzando la clausola **CHECK** (a parte vincoli come UNSIGNED che non la richiedono). Purtroppo essa **è ignorata in MySQL InnoDB** (il DBMS utilizzato) per cui essi non hanno nessun effetto come verrà mostrato più avanti in questo paragrafo.
E' necessario pertanto effettuare i controlli a livello del Web Server (Apache) attraverso il linguaggio di scripting utilizzato (PHP nel nostro caso) ed eventualmente del Client(il browser).

Riassumendo: i vincoli di dominio impostano delle limitazioni per i valori che possono essere assunti da un singolo attributo.
Possono essere impostati attraverso le seguenti clausole:

NOT NULL: se presente richiede che il corrispondente attributo debba necessariamente avere un valore e quindi non possa rimanere “non specificato”.

Es. Cognome varchar(30) NOT NULL

DEFAULT <Valore Predefinito> : assegna un valore predefinito(di default) all'attributo nel caso in cui non ne venga specificato uno

Es. Cognome varchar(30) DEFAULT 'Cognome Sconosciuto'
Stipendio integer(5) DEFAULT 0

CHECK(< condizione>) : serve a specificare un vincolo più specifico sui valori ammissibili

All'interno di CHECK è possibile utilizzare oltre agli operatori di confronto (=, >,<, <=, >= ,

<>) anche i seguenti operatori IN, BETWEEN, NOT BETWEEN, LIKE, NOT LIKE oltre ovviamente gli operati logici AND,OR, NOT

Esempi:
CHECK(Mezzo di trasporto IN ('Autobus','Tram','Treno','Macchina','Bicicletta', 'Nessuno'))
CHECK(Stipendio BETWEEN 1500 AND 3000)
CHECK(Codice Articolo LIKE 'Cod%')

La clausola CHECK può comparire su una riga separata(in genere dopo l'elenco degli attributi) oppure sulla stessa riga dell'attributo su cui è imposto il vincolo (vedi esempi seguenti e le create riportate su questa pagina **esempio**).

Di seguito è riportato l'esempio presente su **w3schools.com**

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CHECK (P_Id > 0)    vincolo di dominio sulla chiave primaria P_Id
)
```

Tuttavia sarebbe possibile teoricamente specificare un vincolo sulla lunghezza del campo **FirstName** nel seguente modo:

CHECK(LENGTH(FirstName) > 3)

dove **LENGTH** è una funzione che restituisce la lunghezza di una stringa

nel seguente esempio è imposto il vincolo sull'età

```
CREATE TABLE Employee (
  Name    VARCHAR(50) PRIMARY KEY NOT NULL,
  PhoneNo VARCHAR(15) DEFAULT 'Unknown Phone',
  Age     INT   CHECK (Age BETWEEN 20 and 30)
);
```

Se proviamo ad eseguire la seguente query:

INSERT INTO Employee (Name, **Age**) VALUES ('John Doe', **31**)

anche se il valore assegnato ad Age è al di fuori dell'intervallo consentito non viene segnalato alcun errori e i dati vengono inseriti proprio perchè i vincoli espressi dalla clausola CHECK non vengono controllati.

VINCOLI DI ENNUPLA(SU GRUPPI DI ATTRIBUTI)

I vincoli di ennupla riguardano più attributi:

Esempi:

PRIMARY KEY (Nome, Cognome, DataNascita, Provincia) è un vincolo che indica che gli attributi specificati tra parentesi formano la chiave primaria della tabella(per ragioni di efficienza e di semplicità, nonché per problemi che potrebbero insorgere dall'utilizzo di chiavi composte è preferibile utilizzare sempre un numero identificativo autoincrementante in questi casi ovvero un ID da aggiungere all'elenco degli attributi).

UNIQUE(CodAzienda,Ragione Sociale) : questo vincolo specifica che non vi possono essere nella stessa tabella due coppie identiche (ovvero con gli stessi valori per i due attributi)

CHECK (StipendioLordo – StipendioNetto = Trattenute) : questo vincolo lega i valori di 3 attributi(StipendioLordo,StipendioNetto, Trattenute) attraverso una relazione matematica