# Progetto Basi di Dati: Database Ferramenta

Daniele Vergara, N46001562
Facoltà di Ingegneria Informatica
Università "Federico II" di Napoli

## Indice



## Introduzione e specifiche sui dati

Si voglia progettare una base di dati per una piccola attività; nello specifico, in questo caso, trattasi di una ferramenta. La base di dati deve fare in modo da semplificare la vita quotidiana del proprietario e dei suoi dipendenti, sostituendo il vecchio sistema informativo e costruendo un solido sistema informatico a supporto del tutto. Essa deve ovviamente rispettare le proprietà ACID (Atomicità, Consistenza, Isolamento, Persistenza) e i dati stessi devono essere prontamente accessibili qualora ve ne sia il bisogno.
Le informazioni conservate devono essere quelle usuali di un esercizio quale la ferramenta, ovvero, il database deve memorizzare quelle relative ai:

• *Dipendenti*, che devono avere un determinato codice di riconoscimento, una mansione a cui sono assegnati, il progetto a cui partecipano (se affidato), un nome, un cognome ed una data di nascita.

• *Materiali*, che si dividono in materiali posseduti e in materiali richiesti/in arrivo. Tutti i materiali devono possedere l'informazione sul tipo di materiale, la quantità ed il fornitore che attualmente lo procura. Sui materiali posseduti è bene sapere in che progetto sono attualmente utilizzati (se

lo sono); dei materiali in arrivo, invece, si devono conservare i dati sulla data di arrivo e sulla data di spedizione.

• *Macchinari*, sui quali si devono memorizzare le informazioni sul codice della macchina, sul tipo della macchina, del dipendente che abitualmente lo usa e del progetto in cui è impegnato (se lo è).

• *Progetti*, i quali devono poter essere facilmente riconosciuti attraverso un codice identificativo, dei quali si conserva l'informazione sulla data di inizio, sulla data (prevista) di conclusione, sul nome (tipo) del progetto, sui materiali utilizzati, sulle macchine utilizzate (se il progetto necessita l'utilizzo di specifiche apparecchiature) e sui dipendenti che sono impegnati nello stesso.

• *Clienti*, dei quali si devono memorizzare i dati sul nome, cognome e sul progetto che ha richiesto (se lo ha fatto).

• *Fornitori*, che devono possedere un nome ed il materiale che attualmente forniscono.

• *Vendite*, delle quali si vuole salvare l'oggetto venduto, a quale cliente, a che prezzo ed in quale data.

Inoltre, esistono 4 categorie di utenti che devono interagire con la base di dati: il database administrator, l'amministrativo, il dipendente ed il cassiere. Per adesso li lasciamo così come sono, ne daremo in seguito una definizione precisa.

# 1 La progettazione concettuale

## 1.1 Analisi delle entità e interconnessioni fra loro

Per progettare una base di dati la prima cosa da fare è sicuramente quella di delineare quali sono le entità cardine che ne fanno parte (in questo caso le abbiamo dalle specifiche), valutare come sono interconnesse fra loro e stilare il cosiddetto diagramma entità/relazione, fornendo così quello che sarà il primo assaggio della base di dati che ne verrà fuori. Nel nostro caso, prima di disegnare il diagramma E/R vero e proprio, abbiamo cercato di schematizzare i concetti principali in forma tabellare dando ad ognuno una descrizione più o meno approfondita e cercando di interpretare ogni concetto con quale altro fosse legato, in modo da intuire più o meno quali fossero le associazioni del rispettivo diagramma entità/relazione.
Ne è venuto fuori lo schema in Figura 1.
Dalla figura si evince naturalmente che, per esempio, il dipendente è collegato al progetto e che il materiale richiesto è collegato ad un fornitore. Senza

dubbio, avremo un'associazione fra il dipendente e il progetto e tra il materiale richiesto e il fornitore, nello schema E/R.

| TERMINE | DESCRIZIONE | TERMINI COLLEGATI |
|---|---|---|
| Dipendente | Il dipendente è colui che si occupa della riuscita dell'attività, svolgendo diversi compiti a lui assegnati (ad es. proprietario, cassiere...). | Progetto |
| Materiale Posseduto | Il materiale posseduto è la materia prima che attualmente l'attività possiede, pronta per essere utilizzata per raggiungere uno scopo (come l'ultimazione di un progetto). | Progetto |
| Materiale Richiesto | Il materiale richiesto è quella parte di materiale che non si possiede ancora ma che è necessario affinchè un progetto possa andare in porto. Esso è procurato dai fornitori. | Fornitore |
| Macchinario | Il macchinario è, appunto, una determinata macchina atta ad automatizzare alcuni processi e sostituirsi talvolta alla manualità dell'uomo. Essa è necessaria per accelerare i tempi di conclusione di un qualsiasi incarico. | Progetto, Dipendente |
| Progetto | Il progetto è un qualsiasi compito (ad es. la realizzazione di una targhetta in ottone) che viene richiesto alla ferramenta dal cliente. | Dipendente, Macchinario, Materiale Posseduto, Cliente |
| Cliente | Il cliente è colui che richiede dei servizi all'attività e che quest'ultima deve adempiere. | Progetto |
| Fornitore | Il fornitore è l'ente che provvede a procurare materiale (o talvolta anche macchinari) all'attività. | Materiale Richiesto |
| Vendita | La vendita è la compera di un oggetto (ad es. giravite) da parte di un cliente che non commissiona alcun progetto. | Cliente |

Figura 1: Tabella dei concetti

## 1.2 Il modello E/R Avanzato

Il modello E/R Avanzato fornisce, come il termine suggerisce, una prima modellizzazione della nostra base di dati. E' avanzato poichè esso contiene le generalizzazioni e le specializzazioni (entità padre ed entità figlio) senza che esse siano state ancora risolte, operazione che compiremo successivamente, per tradurre il modello E/R avanzato in un modello E/R definitivo.
Il diagramma E/R avanzato è disegnato in Figura 2.

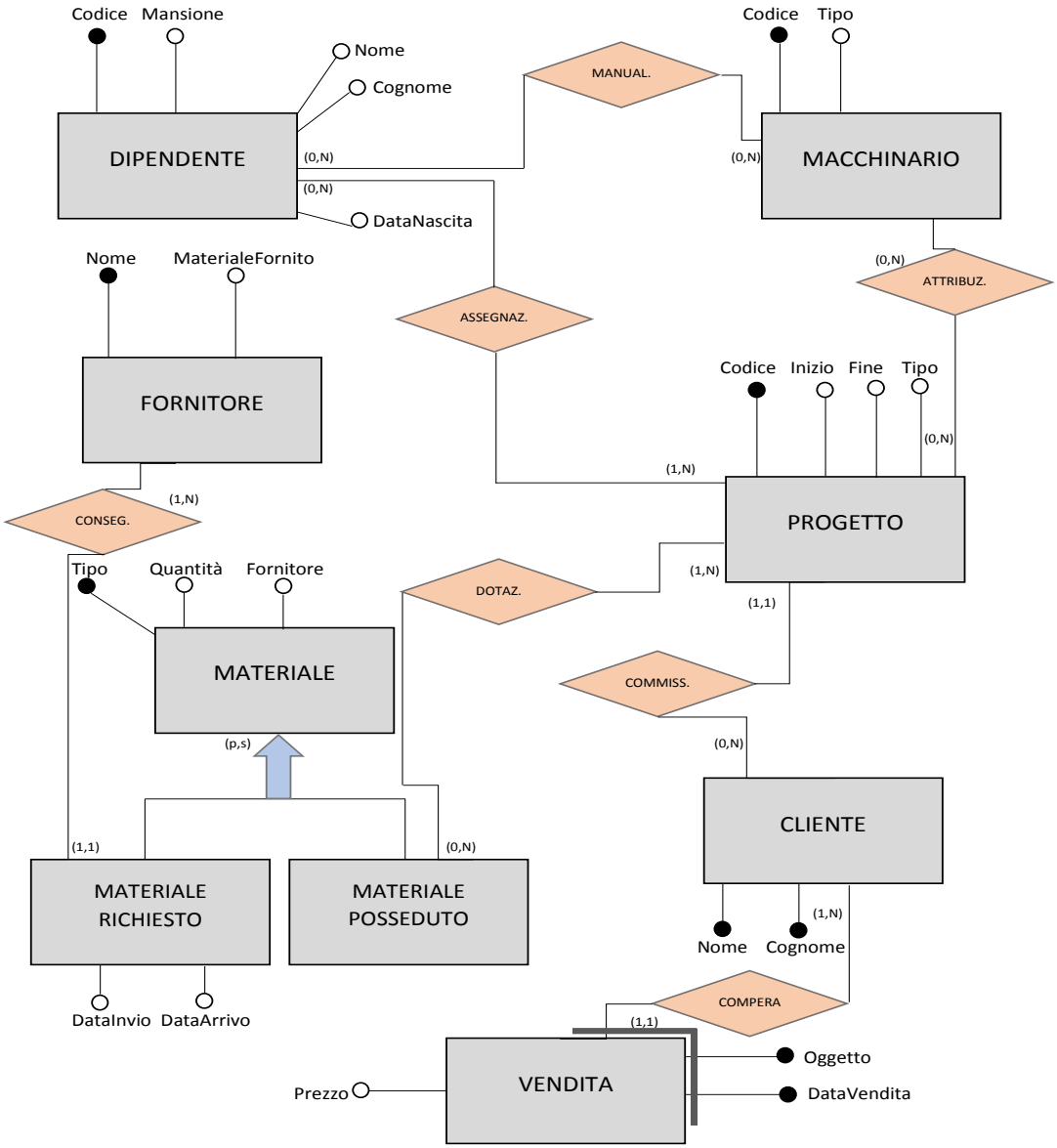


Figura 2: Modello E/R Avanzato

Notiamo dal modello che l'unica specializzazione presente è quella dei materiali, infatti i materiali si dividono - come da specifica - in materiali richiesti e posseduti.
In questo caso, l'entità Materiale è l'entità padre e le due entità figlie sono Materiale Posseduto e Materiale Richiesto.
E' una specializzazione parziale e sovrapposta: parziale perchè non tutti i materiali sono o in proprio possesso o in arrivo, ma potremmo avere anche dei materiali che non toccano minimamente una ferramenta (e che quindi essa non ha nè ordinato nè possiede. Cosa se ne farebbe una ferramenta del petrolio grezzo?).
E' sovrapposta perchè un materiale ordinato può benissimo essere un materiale che l'attività ha già e di cui vuole soltano incrementare la quantià (per esempio, se possediamo già dell'ottone ma ne necessitiamo di altro, lo ordianiamo ugualmente).

## 1.3 Il modello E/R finale

Risolvendo l'unica gerarchia del nostro modello entità/relazione avanzato, siamo in grado di ottenere quello definitivo e che useremo per passare alla progettazione logica della base di dati. Il modello E/R finale è in Figura 3.

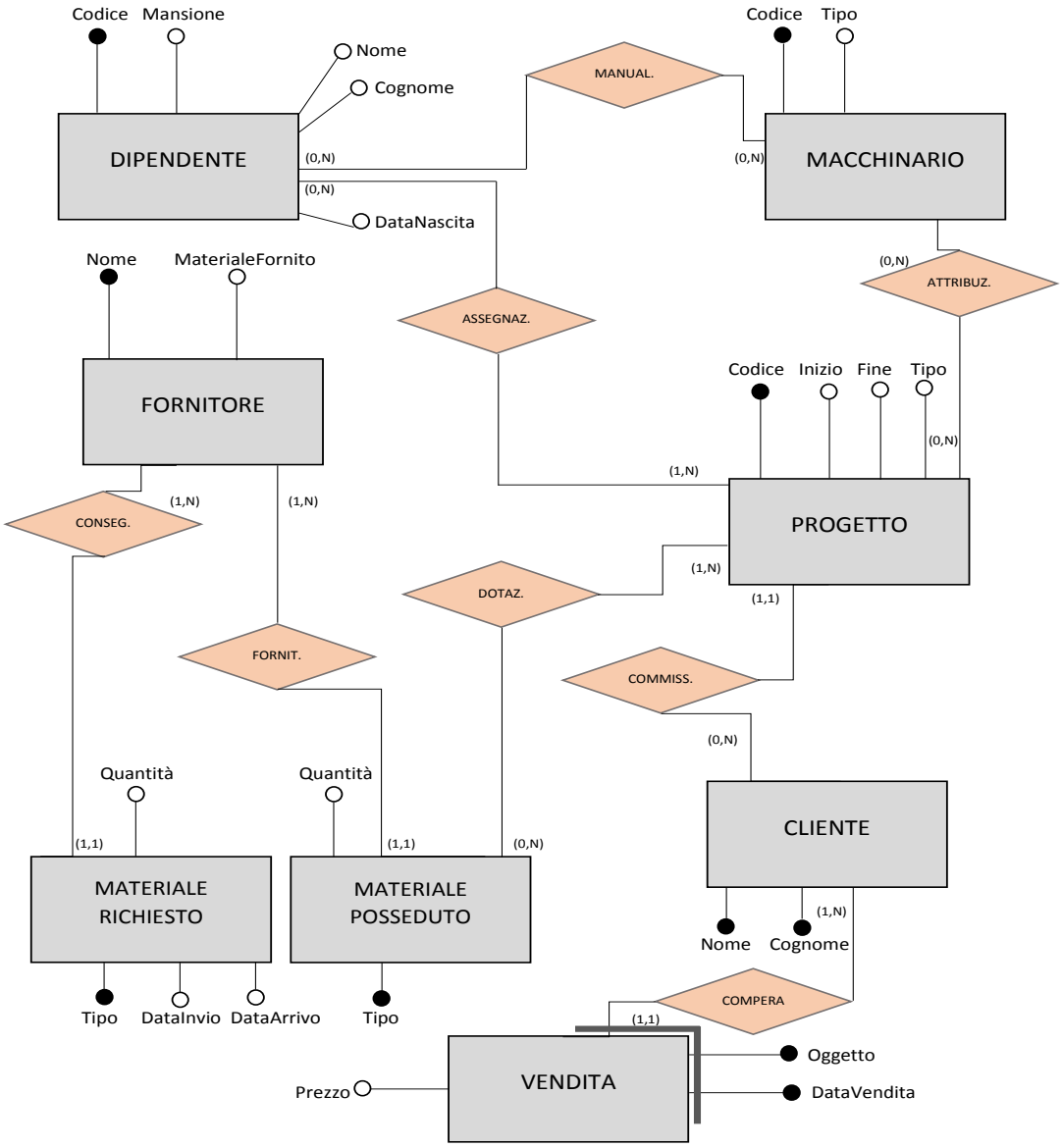


Figura 3: Modello E/R Finale

Abbiamo deciso di risolvere la gerarchia accorpando l'entità padre in ognuna delle due entità figlie, poichè ci è sembrata la scelta più ragionevole da effettuare per semplicità e per efficienza. Abbiamo inoltre aggiunto un'associazione in più rispetto al numero di associazioni presenti nel modello E/R avanzato, ovvero si è scelto di connettere anche l'entità Materiale Posseduto ai Fornitori, in modo da poter rendere l'attributo Fornitori dei materiali posseduti chiave esterna rispetto ai fornitori.

Passiamo alle considerazioni sulla cardinalità attribuita alle varie associazioni tra le entità:

• Un dipendente può lavorare presso nessuno o più macchinari (0,N), mentre un macchinario può lavorare in automatico oppure essere associato a più dipendenti (0,N), ovvero la cardinalità dell'associazione MANUALITA' è di tipo molti a molti.
• Un dipendente può essere assegnato a nessuno o più progetti (0,N), mentre ad un progetto possono partecipare uno o più dipendenti (1,N), ovvero la cardinalità dell'assocazione ASSEGNAZIONE è di tipo uno a molti.
• Un fornitore può consegnare sotto richiesta da uno a più materiali (1,N), mentre il materiale quando richiesto può essere recapitato da uno ed un solo fornitore (1,1), ovvero la cardinalità dell'associazione CONSEGNA è tipo uno a molti.
• Un fornitore può allo stato attuale fornire uno o più materiali alla ferramenta (1,N), mentre il materiale attualmente in possesso può essere stato procurato da uno ed un solo fornitore (1,1), ovvero la cardinalità dell'assocazione FORNITURA è di tipo uno a molti.
• Un macchinario può essere neccessario per sviluppare nessuno o più progetti (0,N), mentre ad un progetto possono essere necessari nessuno o più macchinari per il suo compimento (0,N), ovvero la cardinalità dell'associazione ATTRIBUZIONE è di tipo molti a molti.
• Uno specifico materiale attualmente in possesso della ferramenta può essere utilizzato in nessuno o più progetti (0,N), mentre un progetto può far uso di uno o più materiali al fine di ottenere la sua ultimazione (1,N), ovvero la cardinalità dell'associazione DOTAZIONE è di tipo molti a molti.
• Un cliente può richiedere nessuno o più progetti all'attività (0,N), mentre un progetto può essere commissionato da uno ed un solo cliente della ferramenta (1,1), ovvero la cardinalità dell'associazione COMMISSIONE è di tipo uno a molti.
• Un cliente può effettuare da una a più compere nella ferramenta (1,N), mentre un acquisto può essere effettuato da uno ed un solo acquirente (1,1), ovvero la cardinalità dell'associazione COMPERA è di tipo uno a molti.

Adesso che abbiamo chiare quali sono tutte le associazioni e quali cardinalità hanno, possiamo passare alla fase successiva: la progettazione logica.

# 2 La progettazione logica

La progettazione logica costituisce il punto che presenta la definizione dello schema relazionale della base di dati, con le appropriate relazioni e i vincoli adatti.
Essa si basa sulla trasformazione dello schema concettuale appena ricavato (il modello E/R finale) in un modello relazionale in modo da poter implementare fisicamente il database.

## 2.1 La trasformazione dal modello concettuale a quello relazionale

Utilizzando come riferimento lo schema entità/relazione finale di Figura 3 e prendendo in esame le precedenti considerazioni sulla cardinalità delle associazioni, siamo in grado di completare la trasformazione.
Ovviamente ogni entità diventa a sua volta una relazione, mentre invece per le associazioni utilizziamo le usuali regole di trasformazione delle associazioni, seguendo quelle che sono le loro cardinalità.
Gli identificatori delle entità, inoltre, diventano a loro volta chiavi delle relazioni che andremo a trovare (alcune le rinomineremo adeguatamente).
Trasformando il modello concettuale, otteniamo quello logico:

DIPENDENTI(Codice, Mansione, Nome, Cognome, DataNascita)

MACCHINARI(Codice, Tipo)

FORNITORI(Nome, MaterialeFornito)

MATERIALIRICHIESTI(Tipo, Quantità, DataInvio, DataArrivo, NomeFornitore:FORNITORI)

MATERIALIPOSSEDUTI(Tipo, Quantità, NomeFornitore:FORNITORI)

PROGETTI(Codice, Tipo, Inizio, Fine)

CLIENTI(Nome, Cognome, CodiceProgetto:PROGETTI)

VENDITE(NomeCliente:CLIENTI, CognomeCliente:CLIENTI, Oggetto, DataVendita, Prezzo)

MANUALITA(CodiceDipendente:DIPENDENTI, CodiceMacchinario:MACCHINARI)

ASSEGNAZIONI(CodiceDipendente:DIPENDENTI, CodiceProgetto:PROGETTI)

ATTRIBUZIONI(CodiceMacchinario:MACCHINARI, CodiceProgetto:PROGETTI)

DOTAZIONI(TipoMateriale:MATERIALIPOSSEDUTI, CodiceProgetto:PROGETTI)

Avendo a disposizione ora lo schema relazionale completo per la nostra base di dati, possiamo proseguire fino all'ultima fase: la progettazione fisica.

# 3 La progettazione fisica

La progettazione fisica è il terzo e ultimo step del processo che porterà a compimento la nostra base di dati.
In questa terza parte, come suggerisce il nome stesso, creeremo il vero e proprio database, definiremo le politiche di gestione vere e proprie e i vari vincoli di sicurezza, daremo la possibilià ad user specifici di operare sulla base di dati in una maniera che sarà protetta ed adatta al tipo di user che è attualmente connesso e realizzeremo un'istanza ovvero, procederemo al popolamento della base di dati.

N.B. In questo progetto non tratteremo il dimensionamento della base di dati e daremo per scontato l'hardware e il sistema operativo del server che muoverà il tutto.

## 3.1 Creazione dei ruoli

Come già riportato su e nella prima parte del progetto, uno degli obiettivi che ci siamo posti è quello di rendere il database quanto più sicuro possibile e la prima via per raggiungere lo scopo è quello di creare dei ruoli.
Ciò è a beneficio dell'affidabilità perchè, per esempio, un cassiere della ferramenta che si connette al database per organizzare ciò che gli compete, deve poter modificare, appunto, solo la parte della base di dati che gli spetta, come quella relativa alle vendite. Non ha senso che un cassiere possa vedere

e alterare le informazioni relative ai progetti o possa cambiare la mansione di un dipendente della ferramenta. Per questo, faremo in modo che i cassieri abbiano i privilegi per lavorare solo ed esclusivamente sulla relazione delle vendite, e così via.
Tiriamo fuori il linguaggio SQL e definiamo alcuni dei ruoli più importanti nella nostra attività.

Incominciamo, ovviamente, col dichiarare il database administrator, il quale ha pieno controllo su ogni aspetto del database:

**CREATE USER** FerramentaDBA **IDENTIFIED BY** admin;
**GRANT** dba **TO** FerramentaDBA;

Una volta fatto ciò possiamo proseguire verso i ruoli più "limitati". Andando in ordine decrescente di permessi, definiamo il ruolo "amministrativo". Esso servirà per garantire alcune operazioni su specifiche tabelle ai dipendenti che ricoprono cariche amministrative all'interno della nostra attività. Essi devono poter visualizzare, inserire, modificare, cancellare tuple all'interno delle tabelle relative ai macchinari, ai materiali richiesti, ai materiali posseduti, ai fornitori, ai clienti. Devono inoltre poter gestire la forza lavoro, per cui è necessario che essi possano assegnare dipendenti ad un determinato progetto e/o ad uno specifica apparecchiatura, oltre che a congedarli e/o sostituirli (come di consueto, essi hanno il diritto di visualizzare quali dipendenti sono assegnati ad un macchinario o quali dipendenti stanno o sono in procinto di lavorare ad un progetto particolare).
A tale scopo quindi, esplicitiamo il ruolo:

**CREATE ROLE** Amministrativo;
**GRANT CONNECT TO** Amministrativo;
**GRANT SELECT, INSERT, UPDATE, DELETE ON** FerramentaDBA.MACCHINARI **TO** Amministrativo;
**GRANT SELECT, INSERT, UPDATE, DELETE ON** FerramentaDBA.MATERIALIRICHIESTI **TO** Amministrativo;
**GRANT SELECT, INSERT, UPDATE, DELETE ON** FerramentaDBA.MATERIALIPOSSEDUTI **TO** Amministrativo;
**GRANT SELECT, INSERT, UPDATE, DELETE ON** FerramentaDBA.FORNITORI **TO** Amministrativo;
**RANT SELECT, INSERT, UPDATE, DELETE ON** FerramentaDBA.CLIENTI **TO** Amministrativo;
**GRANT SELECT, INSERT, UPDATE, DELETE ON** FerramentaDBA.MANUALITA **TO** Amministrativo;

**GRANT SELECT, INSERT, UPDATE, DELETE ON** FerramentaDBA.ASSEGNAZIONI **TO** Amministrativo;

Passiamo adesso al secondo ruolo, quello del classico dipendente. Esso deve poter visualizzare, aggiungere, modificare o cancellare dei progetti. I dipendenti devono inoltre poter stabilire quali materiali e quali macchine sono più adeguate ad uno specifico progetto.
Dichiariamo quindi il ruolo Dipendente:

**CREATE ROLE** Dipendente;
**GRANT CONNECT TO** Dipendente;
**GRANT SELECT, INSERT, UPDATE, DELETE ON** FerramentaDBA.PROGETTI **TO** Dipendente;
**GRANT SELECT, INSERT, UPDATE, DELETE ON** FerramentaDBA.ATTRIBUZIONI **TO** Dipendente;
**GRANT SELECT, INSERT, UPDATE, DELETE ON** FerramentaDBA.DOTAZIONI **TO** Dipendente;

Infine, occorre definire il ruolo del cassiere. Il cassiere deve unicamente lavorare sulla tabella vendite come già specificato nell'esempio suddetto, in modo da poter gestire gli acquisti effettuati dai clienti dell'attività.
Quindi:

**CREATE ROLE** Cassiere;
**GRANT CONNECT TO** Cassiere;
**GRANT SELECT, INSERT, UPDATE, DELETE ON** FerramentaDBA.VENDITE **TO** Cassiere;

Procediamo verso la fase successiva.

## 3.2 I create table

Di seguito abbiamo prodotto, in codice SQL, tutte le relazioni che abbiamo ricavato dalla trasformazione dal modello concettuale a quello logico. Senz'altro, ogni attributo avrà i suoi precisi vincoli e il suo specifico tipo (che abbiamo scelto in base al suo adattamento all'attributo stesso).
I comandi che producono le nostre tabelle sono i seguenti:

**CREATE TABLE** DIPENDENTI(
Codice **CHAR(8)**,
Mansione **VARCHAR2(20)**,

Nome **VARCHAR2(25)**,
Cognome **VARCHAR2(25)**,
DataNascita **DATE**,
**PRIMARY KEY**(Codice)
);

**CREATE TABLE** MACCHINARI(
Codice **CHAR(5)**,
Tipo **VARCHAR2(25)**,
**PRIMARY KEY**(Codice)
);

**CREATE TABLE** FORNITORI(
Nome **VARCHAR2(25)**,
MaterialeFornito **VARCHAR2(25)**,
**PRIMARY KEY**(Nome)
);

**CREATE TABLE** MATERIALIRICHIESTI(
Tipo **VARCHAR2(25)**,
Quantità **INTEGER**,
DataInvio **DATE**,
DataArrivo **DATE**,
NomeFornitore **VARCHAR2(25)**,
**PRIMARY KEY**(Tipo),
**FOREIGN KEY**(NomeFornitore) **REFERENCES** FORNITORI(Nome)
**ON DELETE** SET NULL
);

**CREATE TABLE** MATERIALIPOSSEDUTI(
Tipo **VARCHAR2(25)**,
Quantità **INTEGER**,
NomeFornitore **VARCHAR2(25)**,
**PRIMARY KEY**(Tipo),
**FOREIGN KEY**(NomeFornitore) **REFERENCES** FORNITORI(Nome)
**ON DELETE** SET NULL
);

**CREATE TABLE** PROGETTI(
Codice **CHAR(4)**,
Tipo **VARCHAR2(25)**,

Inizio **DATE**,
Fine **DATE**,
**PRIMARY KEY**(Codice)
);
**CREATE TABLE** CLIENTI(
Nome **VARCHAR2(25)**,
Cognome **VARCHAR2(25)**,
CodiceProgetto **CHAR(4)**,
**PRIMARY KEY**(Nome, Cognome),
**FOREIGN KEY**(CodiceProgetto) **REFERENCES** PROGETTI(Codice)
**ON DELETE** SET NULL
);

**CREATE TABLE** VENDITE(
NomeCliente **VARCHAR2(25)**,
CognomeCliente **VARCHAR2(25)**,
Oggetto **VARCHAR2(30)**,
DataVendita **DATE**,
Prezzo **INTEGER**,
**PRIMARY KEY**(NomeCliente, CognomeCliente),
**FOREIGN KEY**(NomeCliente, CognomeCliente) **REFERENCES** CLIEN-
TI(Nome, Cognome)
**ON DELETE** SET NULL
);

**CREATE TABLE** MANUALITA(
CodiceDipendente **CHAR(8)**,
CodiceMacchinario **CHAR(5)**,
**PRIMARY KEY**(CodiceDipendente, CodiceMacchinario),
**FOREIGN KEY**(CodiceDipendente) **REFERENCES** DIPENDENTI(Codice)
**ON DELETE** SET NULL,
**FOREIGN KEY**(CodiceMacchinario) **REFERENCES** MACCHINARI(Codice)
**ON DELETE** SET NULL
);

**CREATE TABLE** ASSEGNAZIONI(
CodiceDipendente **CHAR(8)**,
CodiceProgetto **CHAR(4)**,
**PRIMARY KEY**(CodiceDipendente, CodiceProgetto),
**FOREIGN KEY**(CodiceDipendente) **REFERENCES** DIPENDENTI(Codice)
**ON DELETE** SET NULL,

**FOREIGN KEY**(CodiceProgetto) **REFERENCES** PROGETTI(Codice)
**ON DELETE** SET NULL
);

**CREATE TABLE** ATTRIBUZIONI(
CodiceMacchinario **CHAR(5)**,
CodiceProgetto **CHAR(4)**,
**PRIMARY KEY**(CodiceMacchinario, CodiceProgetto),
**FOREIGN KEY**(CodiceMacchinario) **REFERENCES** MACCHINARI(Codice)
**ON DELETE** SET NULL,
**FOREIGN KEY**(CodiceProgetto) **REFERENCES** PROGETTI(Codice)
**ON DELETE** SET NULL
);

**CREATE TABLE** DOTAZIONI(
TipoMateriale **VARCHAR2(25)**,
CodiceProgetto **CHAR(4)**,
**PRIMARY KEY**(TipoMateriale, CodiceProgetto),
**FOREIGN KEY**(TipoMateriale) **REFERENCES** MATERIALIPOSSE-
DUTI(Tipo)
**ON DELETE** SET NULL,
**FOREIGN KEY**(CodiceProgetto) **REFERENCES** PROGETTI(Codice)
**ON DELETE** SET NULL
);

## 3.3 Popolamento della base di dati

Una volta che abbiamo creato tutte le tabelle della base di dati, passiamo al loro popolamento.
Abbiamo di seguito il codice che inserisce le prime informazioni nelle tabelle che abbiamo generato:

• Insert nella tabella DIPENDENTI
**INSERT INTO** DIPENDENTI **VALUES**('FD451236','Direttore','Dino','Vitale',
to_date('15-03-1950','dd-mm-yyyy'));
**INSERT INTO** DIPENDENTI **VALUES**('FD520101','Contabile','Francesco','Verdi',
to_date('28-10-1967','dd-mm-yyyy'));
**INSERT INTO** DIPENDENTI **VALUES**('AS478569','Tecnico','Antonio','Gianetta',
to_date('02-02-1970','dd-mm-yyyy'));

**INSERT INTO** DIPENDENTI **VALUES**('AS256901','Tecnico','Morgan','Virto',
to_date('10-08-1976','dd-mm-yyyy'));
**INSERT INTO** DIPENDENTI **VALUES**('CS102584','Cassiere','Angela','Tornisco',
to_date('04-05-1979','dd-mm-yyyy'));

• Insert nella tabella MACCHINARI
**INSERT INTO** MACCHINARI **VALUES**('A4563','Cassa');
**INSERT INTO** MACCHINARI **VALUES**('M2032','Pantografo');
**INSERT INTO** MACCHINARI **VALUES**('M4587','Sist. tintometrico');
**INSERT INTO** MACCHINARI **VALUES**('D0035','Duplicatrice');

• Insert nella tabella FORNITORI
**INSERT INTO** FORNITORI **VALUES**('Comitec','Ottone');
**INSERT INTO** FORNITORI **VALUES**('Fision S.r.l.','Plexiglass');
**INSERT INTO** FORNITORI **VALUES**('Tecniq','Alluminio');
**INSERT INTO** FORNITORI **VALUES**('MOTV','Legno');
**INSERT INTO** FORNITORI **VALUES**('Tizen S.r.l.','Plastica');

• Insert nella tabella MATERIALIPOSSEDUTI
**INSERT INTO** MATERIALIPOSSEDUTI **VALUES**('Alluminio',450,'Tecniq');
**INSERT INTO** MATERIALIPOSSEDUTI **VALUES**('Legno',1100,'MOTV');
**INSERT INTO** MATERIALIPOSSEDUTI **VALUES**('Ottone',1500,'Comitec');
**INSERT INTO** MATERIALIPOSSEDUTI **VALUES**('Plexiglass',200,'Fision
S.r.l.');

• Insert nella tabella MATERIALIRICHIESTI
**INSERT INTO** MATERIALIRICHIESTI **VALUES**('Ottone',700,to_date('07-
01-2013','dd-mm-yyyy'),
to_date('11-01-2013','dd-mm-yyyy'),'Comitec');
**INSERT INTO** MATERIALIRICHIESTI **VALUES**('Plastica',300,to_date('20-
12-2012','dd-mm-yyyy'),
to_date('12-01-2013','dd-mm-yyyy'),'Tizen S.r.l.');
**INSERT INTO** MATERIALIRICHIESTI **VALUES**('Alluminio',650,to_date('05-
01-2013','dd-mm-yyyy'),
to_date('13-01-2013','dd-mm-yyyy'),'Tecniq');

• Insert nella tabella PROGETTI
**INSERT INTO** PROGETTI **VALUES**('D4R7','Dupl. chiave',
to_date('05-01-2013','dd-mm-yyyy'),to_date('06-01-2013','dd-mm-yyyy'));
**INSERT INTO** PROGETTI **VALUES**('AS83','Incisione',
to_date('07-01-2013,'dd-mm-yyyy')',to_date('10-01-2013','dd-mm-yyyy'));

**INSERT INTO** PROGETTI **VALUES**('BC40','Tinteggiatura',
to_date('06-01-2013','dd-mm-yyyy'),to_date('15-01-2013','dd-mm-yyyy'));

• Insert nella tabella CLIENTI
**INSERT INTO** CLIENTI **VALUES**('Giuseppe','Arnoldi','AS83');
**INSERT INTO** CLIENTI **VALUES**('Antonio','Russo','BC40');
**INSERT INTO** CLIENTI **VALUES**('Chiara','Romanzi',NULL);
**INSERT INTO** CLIENTI **VALUES**('Emanuela','Di Giacomo','D4R7');

• Insert nella tabella VENDITE
**INSERT INTO** VENDITE **VALUES**('Chiara','Romanzi','Pila',
to_date('04-01-2013,'dd-mm-yyyy'),3);
• Insert nella tabella MANUALITA
**INSERT INTO** MANUALITA **VALUES**('AS478569','M2032');
**INSERT INTO** MANUALITA **VALUES**('AS478569','D0035');
**INSERT INTO** MANUALITA **VALUES**('CS102584','A4563');
**INSERT INTO** MANUALITA **VALUES**('AS256901','M4587');

• Insert nella tabella ASSEGNAZIONI
**INSERT INTO** ASSEGNAZIONI **VALUES**('AS478569','AS83');
**INSERT INTO** ASSEGNAZIONI **VALUES**('AS478569','D4R7');
**INSERT INTO** ASSEGNAZIONI **VALUES**('AS256901','BC40');
**INSERT INTO** ASSEGNAZIONI **VALUES**('FD520101','BC40');

• Insert nella tabella ATTRIBUZIONI
**INSERT INTO** ATTRIBUZIONI **VALUES**('M2032','AS83');
**INSERT INTO** ATTRIBUZIONI **VALUES**('M4587','BC40');
**INSERT INTO** ATTRIBUZIONI **VALUES**('D0035','D4R7');

• Insert nella tabella DOTAZIONI
**INSERT INTO** DOTAZIONI **VALUES**('Alluminio','D4R7');
**INSERT INTO** DOTAZIONI **VALUES**('Ottone','AS83');
**INSERT INTO** DOTAZIONI **VALUES**('Plexiglass','AS83');

## 3.4 Le query sul database

Abbiamo adesso alcuni dati memorizzati nel database della ferramenta: presentiamo qui una serie di interrogazioni alla base di dati al fine di testare la sua risposta di fronte alla necessità di recupero informazioni.

## Query n.1

Trovare il nome e il cognome di tutti i dipendenti della ferramenta:

**SELECT** Nome, Cognome
**FROM** DIPENDENTI

## Query n.2

Trovare il tipo del macchinario il cui codice è "A4563":

**SELECT** Tipo
**FROM** MACCHINARI
**WHERE** Codice="A4563"

## Query n.3

Trovare il tipo di tutti i progetti il cui codice inizia con la lettera D:

**SELECT DISTINCT** Tipo
**FROM** Progetti
**WHERE** Codice LIKE 'D%'

## Query n.4

Trovare i clienti dei quali la seconda lettera del nome è una 'i' e di cui il cognome termina per 'e'. Ordinare poi il risultato in ordine decrescente per cognome (se si trovano due cognomi uguali, allora ordinare per nome):

**SELECT DISTINCT** *
**FROM** CLIENTI
**WHERE** Nome='_i%' **AND** Cognome='%e'
**ORDER BY** Cognome, Nome **DESC**

## Query n.5

Trovare il codice dei progetti a cui sta attualmente partecipando il dipendente con codice "FD520101":

**SELECT** ASSEGNAZIONI.CodiceProgetto
**FROM** DIPENDENTI **JOIN** ASSEGNAZIONI
**ON** DIPENDENTI.Codice=ASSEGNAZIONI.CodiceDipendente
**WHERE** DIPENDENTI.Codice='FD520101'

## Query n.6

Somma delle quantità di tutti i materiali in magazzino:

**SELECT SUM**(Quantità) **AS**
SommaQuantità **FROM** MATERIALIPOSSEDUTI

## Query n.7

Trovare i macchinari che sono impegnati nel progetto "AS83" ma che non sono utilizzati nel progetto "BC40":

**SELECT** M.Codice, M.Tipo
**FROM** MACCHINARI M **JOIN** ATTRIBUZIONI A **ON** M.Codice=A.CodiceMacchinario
**WHERE** A.CodiceProgetto='AS83'
**EXCEPT**
**SELECT** M.Codice, M.Tipo
**FROM** MACCHINARI M **JOIN** ATTRIBUZIONI A **ON** M.Codice=A.CodiceMacchinario
**WHERE** A.CodiceProgetto='BC40'

### Query n.8

Trovare qual è il prezzo medio di vendita dell'oggeto "Pila":

**SELECT AVG**(Prezzo) **AS** MediaPrezzo
**FROM** VENDITE
**WHERE** Oggetti='Pila'

### Query n.9

Materiali in deposito di cui si possiede una quantità superiore alle 1500 unità (sapendo che si possiedono proprio 1500 unità del materiale Ottone):

**SELECT** Tipo
**FROM** MATERIALIPOSSEDUTI
**WHERE** Quantità > **ANY** ( **SELECT** Quantità
**FROM MATERIALIPOSSEDUTI**
**WHERE** Tipo='Ottone' )

### Query n.10

Per ogni progetto, trovare quanti dipendenti ne partecipano:

**SELECT** P.Codice, P.Tipo **AS** Progetto, **COUNT**(A.CodiceDipendente)
**AS** Dipendenti
**FROM** PROGETTO P **JOIN** ASSEGNAZIONI A **ON** P.Codice=A.CodiceDipendente
**GROUP** BY P.Tipo

## 3.5 I Trigger

La creazione dei trigger è l'ultimo passo per l'ultimazione del progetto. I trigger sono fondamentali per automatizzare certi processi (essi matchano il paradigma E-C-A) ed è bene quindi specificati.
La nostra base di dati prevede i seguenti trigger (DML):

## Trigger n.1

Il prezzo di un oggetto non può essere negativo.

1: create or replace trigger prezzononminore
2: before insert on VENDITE
3: for each row
4: declare
5: errore exception;
6: begin
7: if :new.Prezzo < 0
8: then raise errore;
9: end if;
10: exception
11: when errore then raise_application_error(-20002,'Non puoi inserire un prezzo che sia negativo.')
12: end;

## Trigger n.2

Prevediamo un meccanismo che mandi un messaggio di alert ogniqualvolta c'è scarsa disponibilità di un materiale (minore di 100).

1: create or replace trigger materialescarso
2: after insert or update on MATERIALIPOSSEDUTI
3: for each row
4: declare
5: messaggio exception;
6: begin
7: if :new.Quantità < 100
8: then raise messaggio;
9: end if;
10: exception
11: when messaggio then raise_application_error(-20002,'Attenzione, c'è scarsità di materiale!')
12 end;