SAPIENZA
UNIVERSITÀ DI ROMA

Basi di Dati, Modulo 2

Sapienza Università di Roma
Facoltà di Ing. dell'Informazione, Informatica e Statistica
Laurea in Informatica
Prof. Toni Mancini
http://tmancini.di.uniroma1.it

Esercitazione B.3.1.2.3.6 (E.B.3.1.2.3.6)

Basi di Dati Relazionali

La Fase di Progettazione

Progettazione della Base Dati

Produzione dello Schema Relazionale con Vincoli

Traduzione Diretta del Diagramma ER Ristrutturato

Officine 4

– Solo Testo –

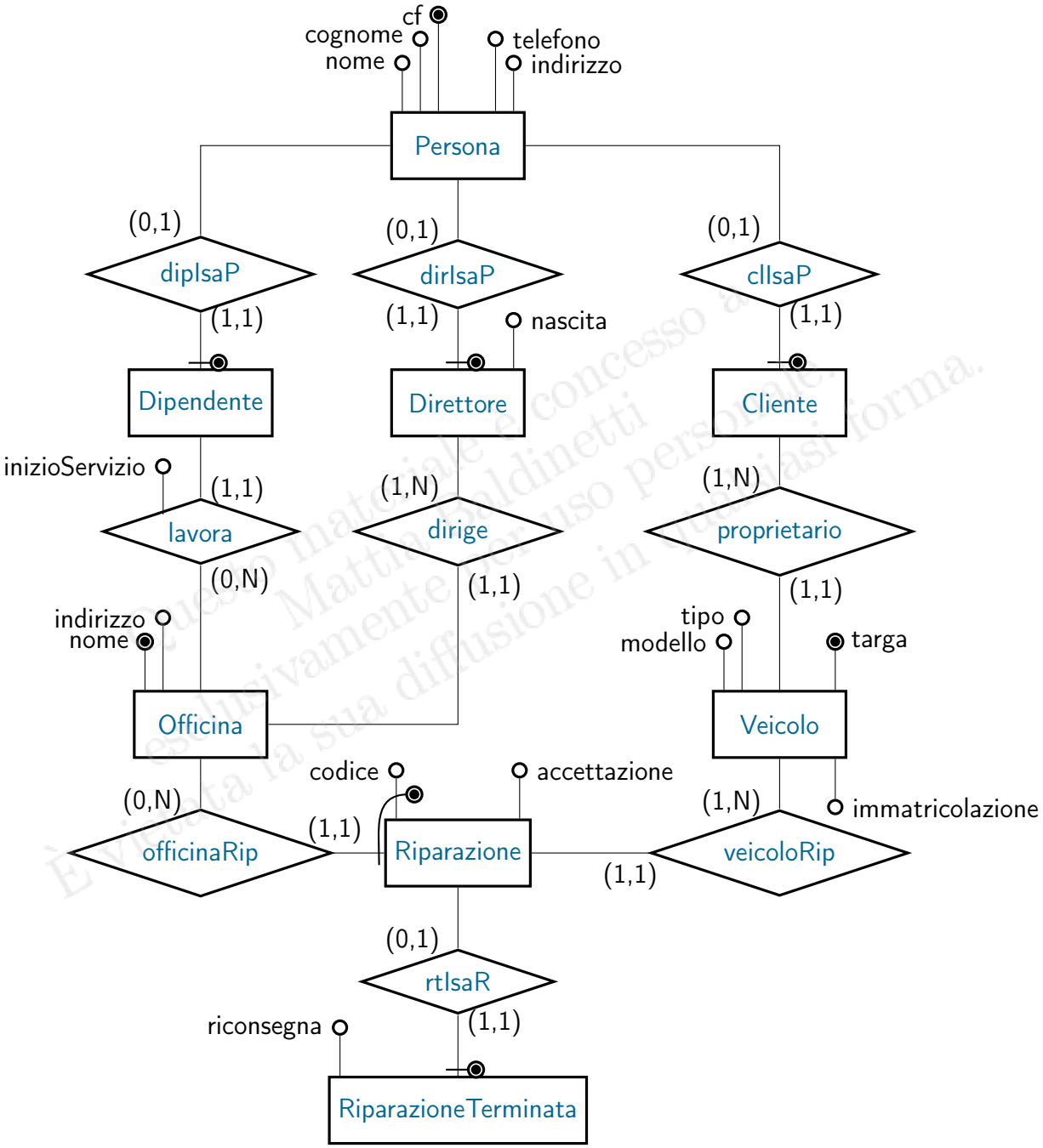
Versione 2019-05-16

Obiettivi

Con riferimento al sistema relativo all'esercitazione "E.B.3.1.2.2.9 – Officine 3", si produca lo schema relazionale della base dati, includendo la definizione dei vincoli che possono essere espressi come vincoli di chiave, chiave primaria, foreign key, ennupla, dominio e di inclusione.

1 Output del Passo di Ristrutturazione

1.1 Diagramma ER Ristrutturato



1.2 Specifiche dei Dati Ristrutturate

Entità Officina

Ogni istanza di questa entità rappresenta una officina della catena.

attributo	dominio	molteplicità	descrizione
nome	StringM		Il nome dell'officina
indirizzo	Indirizzo		L'indirizzo dell'officina

Entità Persona

Ogni istanza di questa entità rappresenta una persona.

attributo	dominio	molteplicità	descrizione
nome	StringM		Il nome della persona
cognome	StringM		Il cognome della persona
cf	CodiceFiscale		Il codice fiscale della persona
indirizzo	Indirizzo		L'indirizzo della persona
telefono	Telefono		Il numero di telefono della persona

Entità Dipendente

Ogni istanza di questa entità rappresenta un dipendente di una officina.
Attributi: Nessuno

Entità Direttore

Ogni istanza di questa entità rappresenta un direttore di una officina.

attributo	dominio	molteplicità	descrizione
nascita	date		La data di nascita del direttore

Vincoli:

[V.Direttore.nonDipendente] Gli insiemi dei direttori e quello dei dipendenti sono disgiunti:

$$\forall p\ \text{Persona}(p) \rightarrow (\exists dir\ \text{dirIsaP}(dir, p) \rightarrow \nexists dip\ \text{dipIsaP}(dip, p))$$

Entità Cliente

Ogni istanza di questa entità rappresenta un cliente di una officina.
Attributi: Nessuno

Entità Veicolo

Ogni istanza di questa entità rappresenta un veicolo registrato nel sistema.

attributo	dominio	molteplicità	descrizione
modello	StringM		Il modello del veicolo
tipo	TipoVeicolo		Il tipo del veicolo
targa	StringS		La targa del veicolo
immatricolazione	IntegerGZ		L'anno di immatricolazione del veicolo

Entità Riparazione

Ogni istanza di questa entità rappresenta la riparazione di un veicolo presso una officina.

attributo	dominio	molteplicità	descrizione
codice	integer		Il codice della riparazione
accettazione	timestamp		La data e l'ora di accettazione del veicolo in riparazione

Entità RiparazioneTerminata

Ogni istanza di questa entità rappresenta la riparazione già terminata di un veicolo presso una officina.

attributo	dominio	molteplicità	descrizione
riconsegna	timestamp		La data e l'ora di riconsegna del veicolo al termine della riparazione

Vincoli:

[V.RiparazioneTerminata.dataora]

$$\forall r, rt, ric, acc\ \text{RiparazioneTerminata}(rt) \wedge \text{Riparazione}(r) \wedge \text{rtISAr}(rt, r) \wedge \text{riconsegna}(rt, ric) \wedge \text{accettazione}(r, acc) \rightarrow acc < ric.$$

Relationship dipIsaP

Ogni istanza di questa relationship lega una istanza di Dipendente alla relativa istanza di Persona
Attributi: Nessuno

Relationship dirIsaP
Ogni istanza di questa relationship lega una istanza di Direttore alla relativa istanza di Persona
Attributi: Nessuno

Relationship clIsaP
Ogni istanza di questa relationship lega una istanza di Cliente alla relativa istanza di Persona
Attributi: Nessuno

Relationship rtIsaR
Ogni istanza di questa relationship lega una istanza di RiparazioneTerminata alla relativa istanza di Riparazione
Attributi: Nessuno

Relationship lavora
Ogni istanza di questa relationship lega un dipendente alla officina presso cui lavora

attributo	dominio	molteplicità	descrizione
inizioServizio	IntegerGEZ		L'anno in cui del dipendente ha iniziato il servizio presso l'officina

Relationship dirige
Ogni istanza di questa relationship lega un direttore alla officina che dirige
Attributi: Nessuno

Relationship proprietario
Ogni istanza di questa relationship lega un veicolo al cliente suo proprietario
Attributi: Nessuno

Relationship veicoloRip
Ogni istanza di questa relationship lega una riparazione al relativo veicolo
Attributi: Nessuno

Vincoli:

[V.veicoloRip.stessoVeicolo]

$$\forall v, r_1, r_2, acc_1, acc_2$$
$$Veicolo(v) \land Riparazione(r_1) \land Riparazione(r_2) \land$$
$$veicoloRip(v, r_1) \land veicoloRip(v, r_2) \land r_1 \neq r_2 \land$$
$$accettazione(r_1, acc_1) \land accettazione(r_2, acc_2) \rightarrow$$
$$\nexists t \ \ dataora(t) \land$$
$$(t \geq acc_1 \land (\forall rt_1, ric_1 \ rtIsaR(rt_1, r_1) \land riconsegna(rt_1, ric_1) \rightarrow t \leq ric_1)) \land$$
$$(t \geq acc_2 \land (\forall rt_2, ric_2 \ rtIsaR(rt_2, r_2) \land riconsegna(rt_2, ric_2) \rightarrow t \leq ric_2))$$

Relationship officinaRip

Ogni istanza di questa relationship lega una riparazione alla officina che l'ha effettuata
Attributi: Nessuno

Dominio CodiceFiscale

Il dominio sarà definito mediante seguente comando SQL:

```
create domain CodiceFiscale as char(16)
    check (isCodiceFiscale(value));
```

dove isCodiceFiscale (**char**(16)): boolean è una opportuna funzione di DB (il cui progetto è lasciato per esercizio) che verifica che il parametro attuale soddisfi i vincoli dei codici fiscali italiani.

Dominio Telefono

Il dominio sarà definito mediante i seguenti comandi SQL:

```
create domain TelefonoCodicePaese as char(5)
    check (value is not null and value ~ '^[0-9]*$');

create domain TelefonoNumero as char(15);}
    check (value is not null and value ~ '^[0-9]*$');

create type Telefono as
    (codicePaese TelefonoCodicePaese,
     numero TelefonoNumero);
```

Dominio Indirizzo

Il dominio sarà definito mediante i seguenti comandi SQL:

```
create domain IndirizzoVia as StringM
    check (value is not null);

create domain IndirizzoCivico as integer
```

```
    check (value > 0);

create domain IndirizzoCAP as char(5)
    check (value is not null and value ~ '^[0-9]*$');

create domain IndirizzoCitta as StringM
    check (value is not null);

create domain IndirizzoNazione as StringM
    check (value is not null);

create type Indirizzo as
    (via IndirizzoVia,
     civico IndirizzoCivico,
     cap IndirizzoCAP,
     citta IndirizzoCitta,
     nazione IndirizzoNazione);
```

Dominio TipoVeicolo

Il dominio sarà definito mediante seguente comando SQL:

```
create type TipoVeicolo as enum
    ('auto', 'moto', 'furgone', 'camion');
```

Dominio StringS

Il dominio sarà definito mediante seguente comando SQL:

```
create domain StringS as varchar(50);
```

Dominio StringM

Il dominio sarà definito mediante seguente comando SQL:

```
create domain StringM as varchar(200);
```

Dominio IntegerGZ

Il dominio sarà definito mediante seguente comando SQL:

```
create domain IntegerGZ as integer check (value > 0);
```

Dominio IntegerGEZ

Il dominio sarà definito mediante seguente comando SQL:

```
create domain IntegerGEZ as integer check (value >= 0);
```

```
create table Persona (
    cf CodiceFiscale not null,
    nome StringM not null,
    cognome StringM not null,
    telefono Telefono not null,
    indirizzo Indirizzo not null,
    primary key (cf)
);
```

accorpa le relationship lavora e dipIsaP

```
create table Dipendente(
    persona CodiceFiscale,
    officina StringM
    inizioServizio IntegerGEZ
    primary key (Persona),
    foreign key: persona references Persona(cf),
    foreign key: officina references Officina(nome)
);
```

la relationship accorpa dirIsaP

```
create table Direttore(
    nascita date not null,
    persona CodiceFiscale,
    primary key (Persona),
    foreign key: persona references Persona(cf),
    inclusione: persona ⊆ Officina(direttore) [dirige]
);
```

la relationship accorpa clIsaP

```
create table Cliente(
    Persona CodiceFiscale,
    primary key (Persona),
    foreign key: persona references Persona(cf),
    inclusione: persona ⊆ Veicolo(proprietario)
);
```

la relationship accorpa dirige

```
create table Officina(
    nome StringM not null,,
    indirizzo Indirizzo not null,
    direttore CodiceFiscale,
    primary key (nome),
    foreign key: direttore references Direttore(persona)
);
```

la relationship accorpa proprietario

```
create table Veicolo(
    targa StringS not null,
    immatricolazione IntegerGZ not null,
    tipo TipoVeicolo not null,
    modello  StringM not null,
    cliente CodiceFiscale,
    primary key (targa),
    foreign key: cliente references Cliente(persona),
    inclusione: targa references Riparazione(veicolo));
```

la relationship accorpa officinaRip e veicoloRip

```
create table Riparazione(
    codice integer,
    officina StringM,
    accettazione timestamp not null,
    veicolo StringS,
    primary key (codice, officina),
    foreign key: officina references Officina(nome)
    foreign key: veicolo references Veicolo(targa)
);


create table RiparazioneTerminata(
    codice integer,
    officina StringM,
    riconsegna datetime not null
    primary key (officina,codice)
    foreign key: (officina,codice) references
Riparazione(officina,codice)
);
```