Gestione impianto produttivo e logistico per Babbo Natale

Progetto per l'insegnamento di Basi di dati 90107

Gabriele Crestanello
#970352
gabriele.crestanello@studio.unibo.it

Mattia Girolimetto
#977478
mattia.girolimetto@studio.unibo.it

Dicembre 2022

Indice

Capitolo 1

Analisi dei requisiti

1.1 Prima intervista

1.1.1 Requisiti iniziali

Babbo Natale vuole digitalizzare la sua impresa e chiede il nostro aiuto per la progettazione della base di dati.

Per consegnare i regali in tutto il mondo Babbo ha assunto dei "babbi delegati" ed ha assegnato a ciascuno di loro un paese di competenza, una slitta e relative renne per fare le consegne. Ad ogni bambino della "lista dei bimbi buoni" si consegnano uno o più regali. Ciascun regalo viene prodotto interamente ed esclusivamente da un solo elfo manifatturiero. Di questi si vuole tenere traccia dei dati personali e contrattuali (es. nome, cognome, stipendio, ...) e a quali regali è stato assegnato. Si vuole inoltre tenere traccia del processo di costruzione di ciascun regalo (da fare, in costruzione o fatto). Per costruire un regalo sono necessari dei materiali (legno, ferro, chiodi, ...), acquistati ciascuno da un fornitore fidato. Per conservare i materiali sono adibiti vari magazzini, ciascuno dei quali è destinato ad una o più categorie di materiali (es il magazzino 1 ha legno e chiodi, il magazzino 2 ha ferro e polistirolo e colla...). Si vuole sapere in ogni momento la quantità di materiale rimasto. L'impianto produttivo è diviso in officine, ciascuna identificata da un codice univoco formato da una lettera e un numero (es "A13", "D7", ...) il cui significato è noto solo a Babbo Natale. In ciascuna officina lavora un numero arbitrario di elfi e ciascun elfo lavora in una sola officina.

1.1.2 Operazioni richieste

1. Dati nome e cognome di un bambino, restituire i nomi e cognomi dei babbi delegati che gli hanno consegnato i regali.

2. Restituire nome e cognome di tutti gli elfi che hanno prodotto almeno un regalo trasportato da un dato babbo delegato.

3. Restituire il costo per produrre un dato regalo.

4. Ricostruire l'albero genialogico di una data renna.

5. Restituire nomi e cognomi dei bambini presenti nella lista dei bambini correntemente buoni.

6. Trovare nomi e cognomi di tutti i bambini che in passato erano buoni e che correntemente non lo sono più.

7. Restituire lo stock rimanente dei materiali necessari per la costruzione dei regali destinati a bambini italiani.

8. Dato un elfo, restituire l'elenco di tutti i materiali (ordinati per categoria) che ha usato per la costruzione di regali nel corso del tempo.

9. Restituire nomi e cognomi dei bambini che hanno ricevuto una consegna da un determinato modello di slitta nel corso del tempo.

10. Restituire nome, cognome e orario di lavoro di tutti gli elfi attualmente in attività.

11. Restituire i fornitori ordinati in base al valore dei materiali venduti e attualmente in magazzino.

12. Data una renna restituire tutti i suoi fratelli e sorelle.

13. Restituire la somma di tutti gli stipendi che babbo natale paga ai suoi dipendenti.

1.2 Ulteriori interviste

Sono state poste ulteriori domande al cliente per ottenere ulteriori dettagli e risolvere alcune ambiguità. Sono riporate sottostante le conclusioni:

- Ogni categoria di materiale è contenuta *esclusivamente* in un singolo magazzino.
- Le slitte, quando non in uso, vengono parcheggiate in dei garage.
- Garage e magazzini sono identificati da un codice univoco come quello usato per le officine.
- Questo codice univoco è in forma

  $$[A-Z][0-9][0-9]?$$

  ovvero una lettera seguita da una o due cifre.
- Le officine e i magazzini hanno degli orari di apertura e chiusura. Non è quindi sempre possibile accedere a questi edifici.
- Essendo le renne allevate dal personale di Babbo Natale si vuole tenere traccia anche del loro albero genialogico, nonché della loro nascita ed eventuale morte.
- La lista dei bimbi buoni viene aggiornata ogni anno. Se un bimbo è stato buono nel 2022 potrebbe non esserlo nel 2023, e viceversa.
- Quando un babbo delegato viene assunto riceve in affidamento una slitta che userà fino alla scadenza del suo contratto.
- Ogni babbo delegato ha un solo stato di competenza, ma più babbi delegati potrebbero avere assegnato lo stesso stato. Questo per poter consegnare i regali in modo efficente anche in paesi la popolazione è molto numerosa.

A seguito di questa seconda intervista i requisiti non contengono più ambiguità.

1.3 Glossario dei termini

Termine	**Significato**	**Sinonimi**
Babbo delegato	Persona assunta da Babbo Natale per fare le consegne	Delegato
Elfo Manifatturiero	Entità assunta da Babbo Natale per costruire i regali	Elfo
Bambino Buono	Bambino il cui nome è nella lista dei bimbi buoni di Babbo Natale	Bambino
Conesegna	Insieme di regali destinati ad uno stesso bambino buono	-
Magazzino	Edificio in cui si conservano materiali	-
Officina	Edificio in cui gli elfi costruiscono i regali	-
Garage	Edificio in cui vengono conservate le slitte	-
Codice Univoco	Stringa alfanumerica nel formato $[A-Z][0-9][0-9]?$	-

Capitolo 2

Progettazione

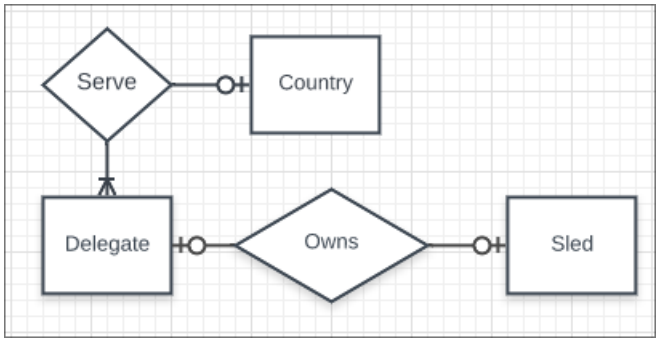
2.1 Progettazione concettuale

2.1.1 Identificare entità, relazioni e attributi

Dopo aver studiato attentamente i requisiti sono state ricavate delle proposizioni utili per identificare entità, relazioni e attributi. Di seguito queste sono elencate affiancate alla loro rappresentazione secondo il modello entità-relazione.

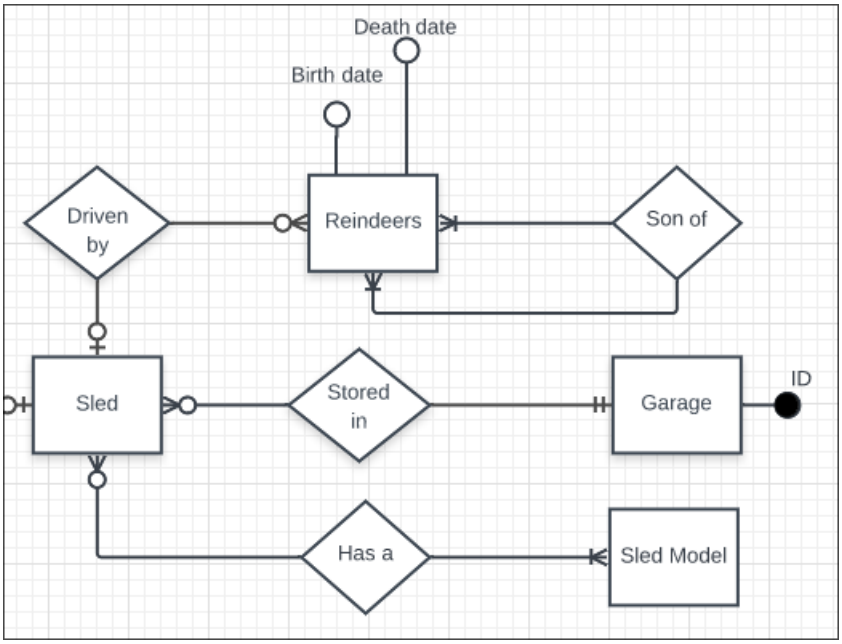
Proposizioni sui Babbi delegati

- Esistono dei babbi delegati
- Esistono degli stati
- Ogni delegato ha uno e uno solo stato di competenza
- Esistono delle slitte
- Ogni delegato ha una sola slitta, che userà per tutta la durata del suo contratto



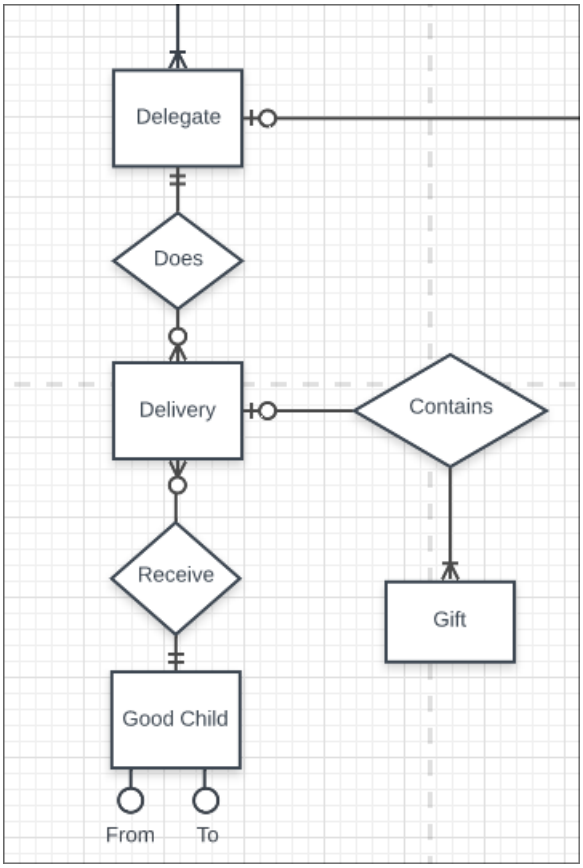
Proposizioni su slitte e renne

- Esistono dei garage
- Le slitte vengono parcheggiate nei garage
- Esistono delle renne
- Ogni renna ha una madre, un padre, una data di nascita e eventualmente di morte
- Ogni slitta ha delle renne
- Ogni garage ha un codice univoco



Proposizioni su bambini e regali

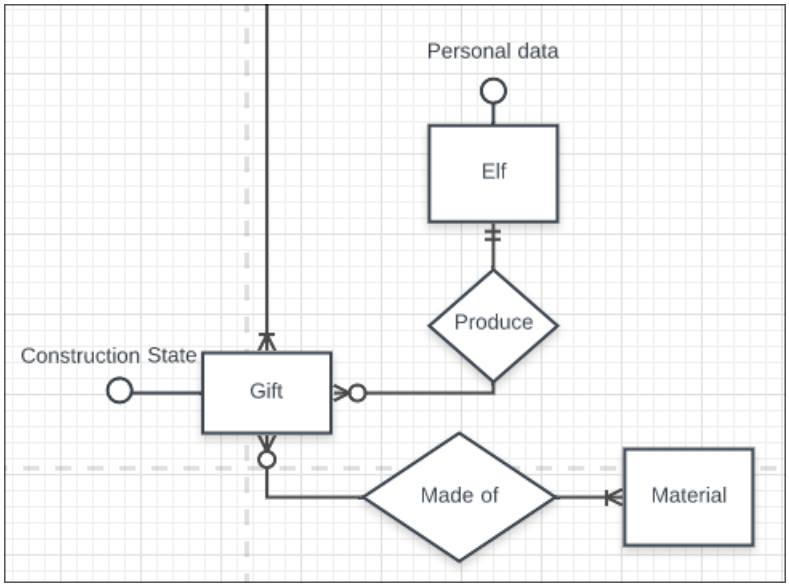
- Esistono dei bambini
- Ogni bambino può essere buono o no
- Esistono dei regali
- Ogni bambino buono ha diritto ad uno o più regali



Proposizioni su elfi e materiali

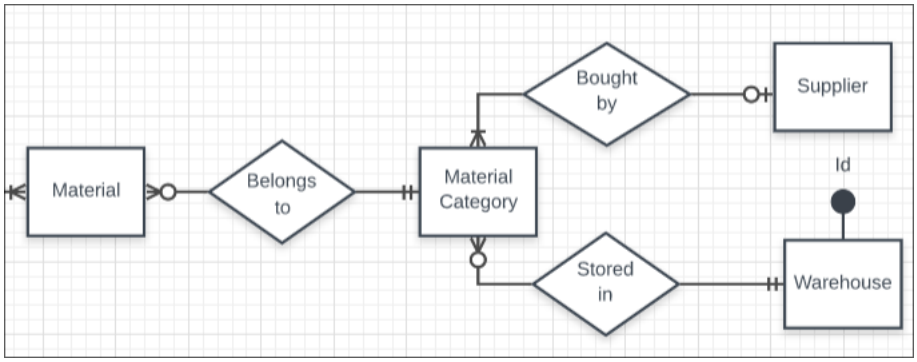
- Esistono degli elfi
- Ogni regalo è prodotto da un singolo elfo
- Ogni elfo ha dei dati personali e contrattuali

- Ogni regalo ha uno stato di costruzione
- Esistono dei materiali
- Ogni regalo è costruito con dei materiali



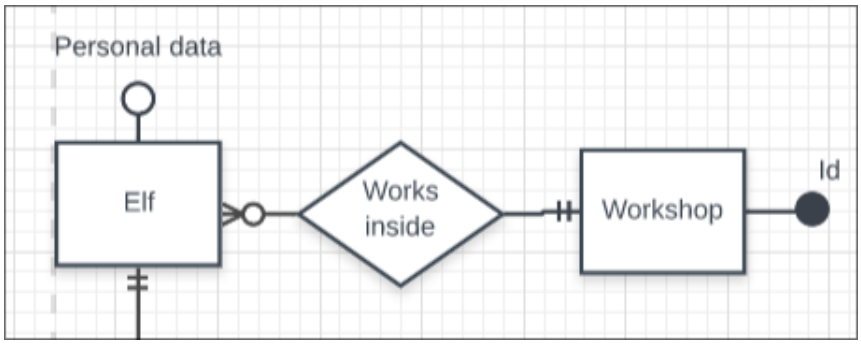
Proposizioni su categorie di materiali e fornitori

- Esistono dei fornitori
- Esistono delle categorie di materiali
- Ogni materiale ha una categoria
- Ogni categoria di materiali è acquistata da un fornitore
- Ogni fornitore ci vende una o più categorie
- Esistono dei magazzini
- Ogni magazzino ha un codice univoco
- Ogni magazzino ha delle categorie di meriali



Proposizioni sulle officine

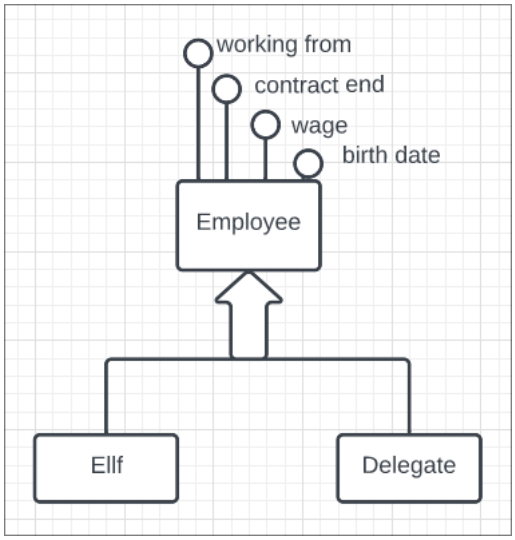
- Esistono delle officine
- Ogni officina ha un codice univoco
- Ogni elfo lavora in un'officina
- Ogni officina ha degli elfi che ci lavorano dentro



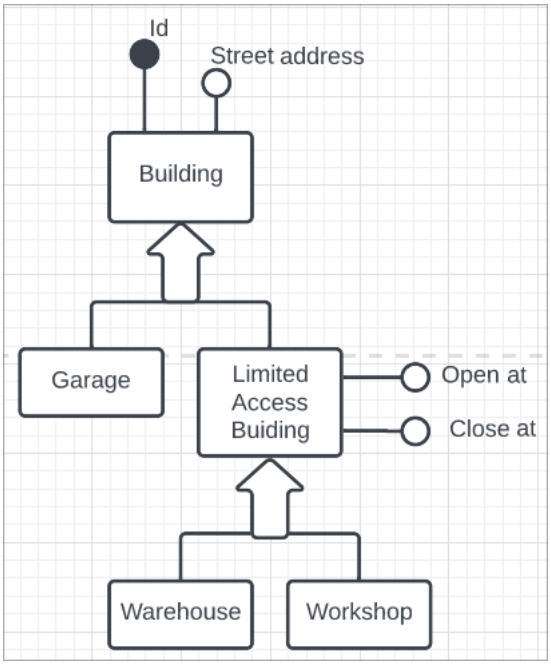
2.1.2 Gerarchie IS-A

Usando questa modellazione dei dati sono state individuate due possibili gerarchie *is-a*.

Impiegati La prima riguarda le entità *Elfo* e *Delegato*. Queste infatti rappresentano due tipologie di impieghi all'interno dell'azienda di Babbo Natale. Per questo possono essere viste come una specializzazione di un'entità generale *Impiegato*, che avrà come attributi, ad esempio, tutte le informazioni del contratto di assunzione.

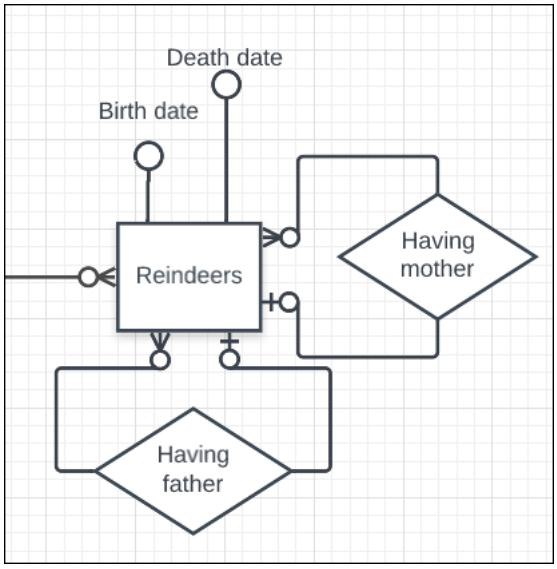


Edifici La seconda realazione *is-a* individuata è quella tra le entità che rappresentano gli edifici, ovvero *garage*, *workshop* e *warehouse*. Questi infatti condividono informazioni comuni come l'indirizzo e il codice univoco (nello stesso formato per tutti e tre). Analizzando più attentamente il problema si nota che *workshop* e *warehouse*, a differenza di *garage*, condividono delle informazioni aggiuntive: quelle riguardanti l'orario di apertura e chiusura dell'edificio. Quindi è stata individuata una doppia relazione *is-a*:



2.1.3 Relazione ricorsiva

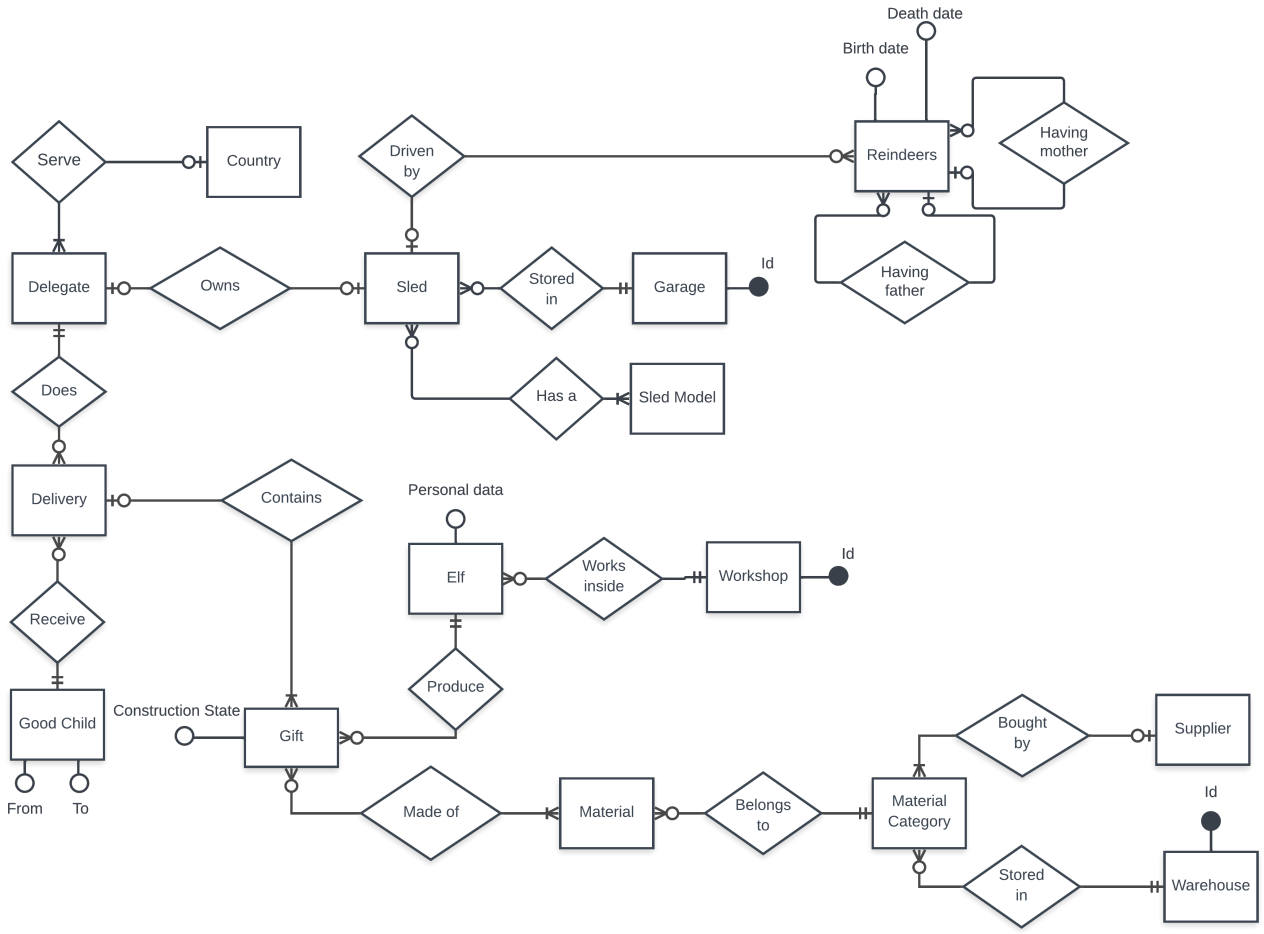
Un' altra osservazione fatta riguarda la relazione ricorsiva *son of* delle entità *reindeers*. Al fine di mantenere una più accurata rappresentazione della realtà e semplificare lo schema, è possibile rimovere la relazione *son of* di tipo *molti a molti* ed inserire due relazioni di tipo *uno a molti*. Queste rappresentano il concetto di parentela madre-figlio e padre-figlio. Lo schema diventa quindi:



Si noti che sono state modificate anche le molteplicità: sebbene è impossibile per natura che una renna non abbia una madre e un padre, si è preferito tenere la relazione opzionale nel caso in cui ci siano renne i cui genitori siano sconosciuti. Nell'altro verso è stata scelta la molteplicità *zero o molti* poiché è ragionevole pensare che una renna possa non avere figli.

2.1.4 Conclusioni

Alla fine della progettazione concettuale lo schema entità per intero è il seguente:



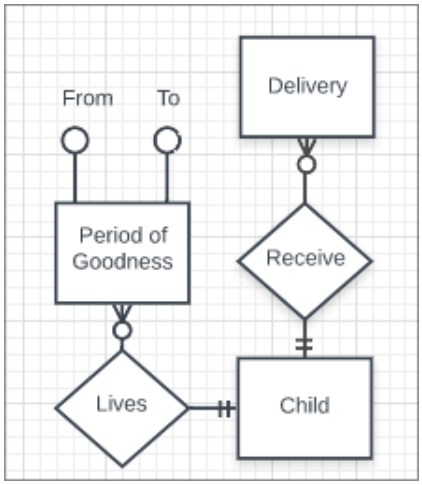
2.2 Progettazione logica

2.2.1 Relazioni molti a molti

Nello schema ER è presente una relazione di tipo *molti a molti* tra le entità *gift* e *material*. Per rappresentarla correttamente nel modello logico è stata aggiunta l'entità intermedia *GiftMaterial* come segue

2.2.2 Il caso dei bambini buoni

Nello schema ER progettato i bambini buoni sono rappresentati da un'entità *Good child* con campi *from* e *to*. Questi servono a dire che il bambino identificato dagli attributi di quel elemento della relazione è *stato buono da <from> a <to>*, dove $< to >$ può assumere valore $NULL$ nel caso in cui il bimbo sia correntemente buono. Questo però implica una notevole forma di ridondanza, in quanto per ogni cambiamento di stato di comportamento del bambino, si deve inserire un nuovo record nella tabella. Per risolvere questa ridondanza si è sostituita l'entità *GoodChild* con *Child* e *PeriodOfGoodness* come riportato in seguito:



2.2.3 Tavole dei volumi e delle operazioni

Sottostante sono riportate le tavole dei volumi e delle operazioni. Si noti che alcune tabelle (regali, spedizioni ad esempio) tendono a crescere parecchio, in modo lineare, col passare degli anni.

Tabella dei volumi

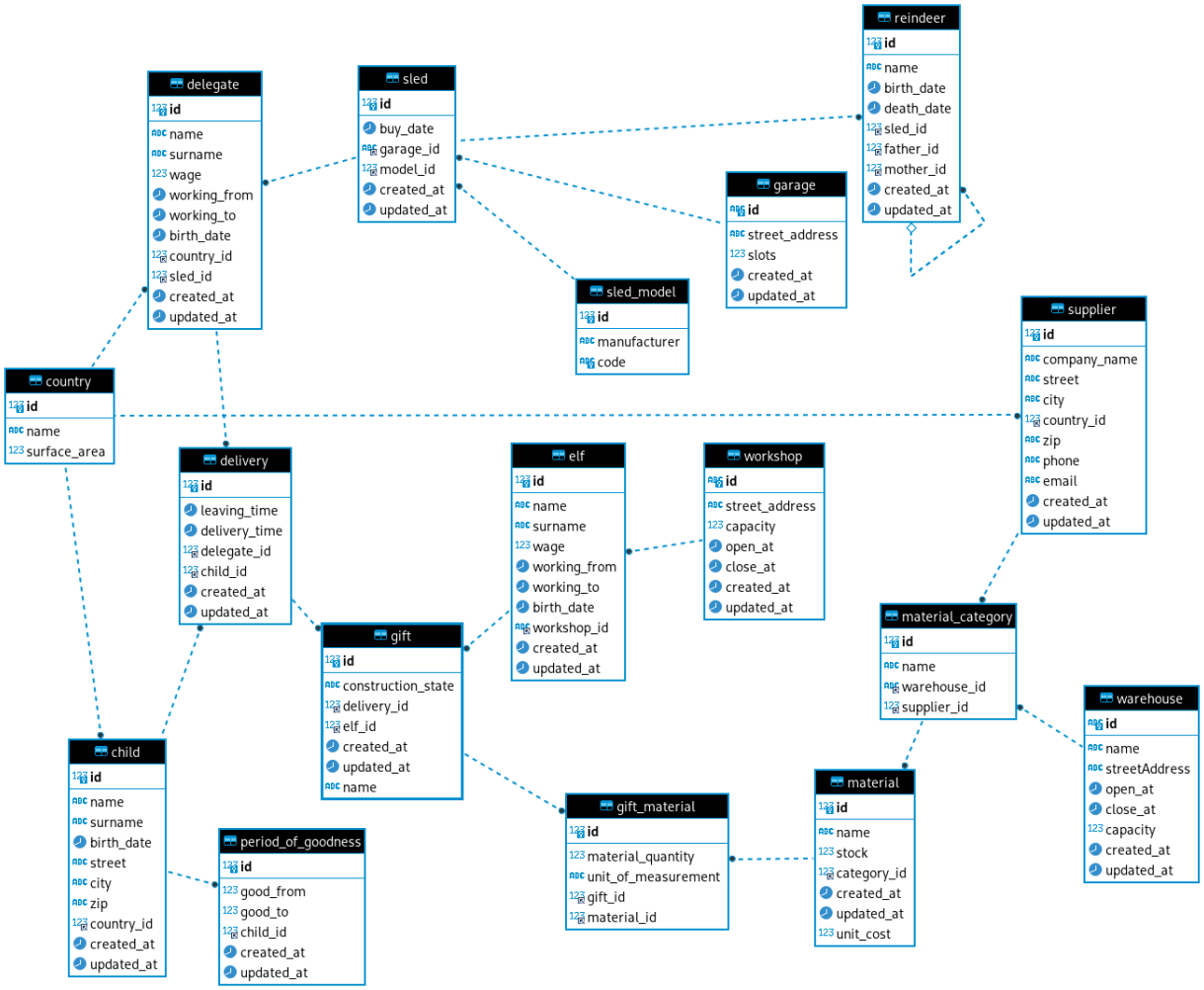
Concetto	**Tipo**	**Volume**
Stato	E	200
Babbi Delegati	E	400
Slitte	E	450
Modelli di slitta	E	50
Garage	E	20
Renne	E	2700
Spedizioni	E	2.200.000.000
Bambini	E	2.200.000.000
Periodi di bontà	E	2.200.000.000
Regali	E	6.600.000.000
Elfi Manifatturieri	E	500.000
Officine	E	1000
Materiali	E	5000
Categorie di materiali	E	500
Fornitori	E	500
Magazzini	E	100

Tabella delle operazioni

Operazione	**Frequenza annuale**
1	1000
2	1000
3	7.000.000.000
4	2700
5	500
6	500
7	500
8	1000
9	500
10	1000
11	1000
12	100
13	2000

2.2.4 Schema logico

Entità	**Attributi**	**Relazioni**
Country	Id, Name, SurfaceArea	
Delegate	Id, Name, Surname, Wage, WorkingFrom, WorkingTo, BirthDate, CountryId, SledId	CountryId → Country.Id SledId → Sled.Id
Sled	Id, GarageId, ModelId, BuyDate	
SledModel	Id, manufacturer, code	
Garage	Id, StreetAddress, Slots	
Reindeer	Id, SledId, Name, BirthDate, DeathDate, FatherId, MotherId	SledId → Sled.Id FatherId → Reindeer.Id MotherId → Reindeer.Id
Delivery	Id, LeavingTime, DeliveryTime, DelegateId, ChildId	DelegateId → Delegate.Id, ChildId → Child.Id
Child	Id, BirthDate, Name, Surname, StreetAddress, City, CountryId, ZIP	CountryId → Country.Id
PeriodOfGoodness	Id, From, To , ChildId	ChildId → Child.Id
Gift	Id, ConstructionState, DeliveryId, ElfId	DeliveryId → Delivery.Id ElfId → Elf.Id
GiftMaterial	Id, MaterialQuantity, UnitOfMeasurement, GiftId, MaterialId	GiftId → Gift.Id MaterialId → Material.Id
Elf	Id, Name, Surname, Wage, WorkingFrom, WorkingTo, BirthDate, WorkshopId	WorkshopId → Workshop.Id
Workshop	Id, Name, StreetAddress, Capacity, OpenAt, CloseAt	
Material	Id, Name, Stock, CategoryId	CategoryId → MaterialCategory.Id
MaterialCategory	Id, Name, WarehouseId, SupplierId	WarehouseId → Warehouse.Id SupplierId → Supplier.Id
Supplier	Id, CompanyName, StreetAddress, City, CountryId, ZIP, Phone, Email	CountryId → Country.Id
Warehouse	Id, Name, StreetAddress, OpenAt, CloseAt, Capacity	



2.2.5 Normalizzazione

E' possibile dimostrare che lo schema logico rappresentato in precedenza è nella forma normale di Boyce-Codd. Ciascuna relazione infatti presenta solo dipendenze funzionali non banali il cui determinante è una chiave o una possibile super chiave.

Capitolo 3

Codifica SQL

3.0.1 Definizione dello schema

```
CREATE type CONS_STATE AS ENUM (
    'to␣do',
    'designing',
    'sculpting',
    'crafting',
    'painting',
    'forging',
    'finished'
);


CREATE TABLE IF NOT EXISTS country (
        id SERIAL PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        surface_area INTEGER NOT NULL
    );

CREATE TABLE IF NOT EXISTS garage (
        id VARCHAR(3) PRIMARY KEY,
        street_address VARCHAR(255) NOT NULL,
        slots SMALLINT NOT NULL,
        created_at TIMESTAMP NOT NULL DEFAULT NOW(),
        updated_at TIMESTAMP NOT NULL DEFAULT NOW()
```

```
    );

CREATE TABLE IF NOT EXISTS sled_model (
        id SERIAL PRIMARY KEY,
        manufacturer VARCHAR(255) NOT NULL,
        code VARCHAR(255) NOT NULL UNIQUE
    );
CREATE TABLE IF NOT EXISTS sled (
        id SERIAL PRIMARY KEY,
        buy_date TIMESTAMP NOT NULL,
        garage_id VARCHAR(3) NOT NULL,
        model_id INTEGER NOT NULL,
        created_at TIMESTAMP NOT NULL DEFAULT NOW(),
        updated_at TIMESTAMP NOT NULL DEFAULT NOW(),
        FOREIGN KEY (garage_id) REFERENCES garage (id),
        FOREIGN KEY (model_id) REFERENCES sled_model (id)
    );

CREATE TABLE IF NOT EXISTS delegate (
        id SERIAL PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        surname VARCHAR(255) NOT NULL,
        wage REAL NOT NULL,
        working_from DATE NOT NULL DEFAULT NOW(),
        working_to DATE DEFAULT NULL,
        birth_date DATE NOT NULL,
        country_id INTEGER NOT NULL,
        sled_id INTEGER NOT NULL,
        created_at TIMESTAMP NOT NULL DEFAULT NOW(),
        updated_at TIMESTAMP NOT NULL DEFAULT NOW(),
        FOREIGN KEY (country_id) REFERENCES country (id),
        FOREIGN KEY (sled_id) REFERENCES sled (id),
        CHECK (working_from < working_to),
        CHECK (wage > 0)
    );
CREATE TABLE IF NOT EXISTS reindeer (
        id SERIAL PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
```

```
        birth_date DATE NOT NULL,
        death_date DATE DEFAULT NULL,
        sled_id INTEGER NOT NULL,
        father_id INTEGER,
        mother_id INTEGER,
        created_at TIMESTAMP NOT NULL DEFAULT NOW(),
        updated_at TIMESTAMP NOT NULL DEFAULT NOW(),
        FOREIGN KEY (sled_id) REFERENCES sled (id),
        FOREIGN KEY (father_id) REFERENCES reindeer (id),
        FOREIGN KEY (mother_id) REFERENCES reindeer (id),
        CHECK(death_date > birth_date)
    );

CREATE TABLE IF NOT EXISTS child (
        id SERIAL PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        surname VARCHAR(255) NOT NULL,
        birth_date DATE NOT NULL,
        street VARCHAR(255) NOT NULL,
        city VARCHAR(255) NOT NULL,
        zip VARCHAR(255) NOT NULL,
        country_id INTEGER NOT NULL,
        created_at TIMESTAMP NOT NULL DEFAULT NOW(),
        updated_at TIMESTAMP NOT NULL DEFAULT NOW(),
        FOREIGN KEY (country_id) REFERENCES country (id)
    );
CREATE TABLE IF NOT EXISTS period_of_goodness (
        id SERIAL PRIMARY KEY,
        good_from SMALLINT NOT NULL,
        good_to SMALLINT DEFAULT NULL,
        child_id INTEGER NOT NULL,
        created_at TIMESTAMP NOT NULL DEFAULT NOW(),
        updated_at TIMESTAMP NOT NULL DEFAULT NOW(),
        FOREIGN KEY (child_id) REFERENCES child (id),
        CHECK (good_to > good_from)
    );

CREATE TABLE IF NOT EXISTS delivery (
```

```
        id SERIAL PRIMARY KEY,
        leaving_time TIMESTAMP DEFAULT NULL,
        delivery_time TIMESTAMP DEFAULT NULL,
        delegate_id INTEGER NOT NULL,
        child_id INTEGER NOT NULL,
        created_at TIMESTAMP NOT NULL DEFAULT NOW(),
        updated_at TIMESTAMP NOT NULL DEFAULT NOW(),
        FOREIGN KEY (delegate_id) REFERENCES delegate (id),
        FOREIGN KEY (child_id) REFERENCES child (id),
        CHECK (leaving_time < delivery_time)
    );

CREATE TABLE IF NOT EXISTS workshop (
        id VARCHAR(3) PRIMARY KEY,
        street_address VARCHAR(255) NOT NULL,
        capacity INTEGER NOT NULL DEFAULT 0,
        open_at TIME NOT NULL,
        close_at TIME NOT NULL,
        created_at TIMESTAMP NOT NULL DEFAULT NOW(),
        updated_at TIMESTAMP NOT NULL DEFAULT NOW(),
        CHECK (open_at < close_at)
    );
CREATE TABLE IF NOT EXISTS elf (
        id SERIAL PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        surname VARCHAR(255) NOT NULL,
        wage REAL NOT NULL,
        working_from DATE NOT NULL,
        working_to DATE DEFAULT NULL,
        birth_date DATE NOT NULL,
        workshop_id VARCHAR(3) NOT NULL,
        created_at TIMESTAMP NOT NULL DEFAULT NOW(),
        updated_at TIMESTAMP NOT NULL DEFAULT NOW(),
        FOREIGN KEY (workshop_id) REFERENCES workshop (id),
        CHECK (wage > 0),
        CHECK (working_from < working_to)
    );
```

```
CREATE TABLE IF NOT EXISTS gift (
        id SERIAL PRIMARY KEY,
        construction_state CONS_STATE NOT NULL,
        name VARCHAR(100) NOT NULL,
        delivery_id INTEGER NOT NULL,
        elf_id INTEGER NOT NULL,
        created_at TIMESTAMP NOT NULL DEFAULT NOW(),
        updated_at TIMESTAMP NOT NULL DEFAULT NOW(),
        FOREIGN KEY (delivery_id) REFERENCES delivery (id),
        FOREIGN KEY (elf_id) REFERENCES elf (id)
    );

CREATE TABLE IF NOT EXISTS warehouse (
        id VARCHAR(3) PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        streetAddress VARCHAR(255) NOT NULL,
        open_at TIME NOT NULL,
        close_at TIME NOT NULL,
        capacity INTEGER NOT NULL DEFAULT 0,
        created_at TIMESTAMP NOT NULL DEFAULT NOW(),
        updated_at TIMESTAMP NOT NULL DEFAULT NOW(),
        CHECK (open_at < close_at)
    );

CREATE TABLE IF NOT EXISTS supplier (
        id SERIAL PRIMARY KEY,
        company_name VARCHAR(255) NOT NULL,
        street VARCHAR(255) NOT NULL,
        city VARCHAR(255) NOT NULL,
        country_id INTEGER NOT NULL,
        zip VARCHAR(7) NOT NULL,
        phone VARCHAR(15) NOT NULL,
        email VARCHAR(255) NOT NULL,
        created_at TIMESTAMP NOT NULL DEFAULT NOW(),
        updated_at TIMESTAMP NOT NULL DEFAULT NOW(),
        FOREIGN KEY (country_id) REFERENCES country (id)
    );
```

```
CREATE TABLE IF NOT EXISTS material_category (
        id SERIAL PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        warehouse_id VARCHAR(3) NOT NULL,
        supplier_id INTEGER NOT NULL,
        FOREIGN KEY (warehouse_id) REFERENCES warehouse (id),
        FOREIGN KEY (supplier_id) REFERENCES supplier (id)
    );

CREATE TABLE IF NOT EXISTS material (
        id SERIAL PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        stock INTEGER NOT NULL DEFAULT 0,
        category_id INTEGER NOT NULL,
        unit_cost INTEGER NOT NULL,
        created_at TIMESTAMP NOT NULL DEFAULT NOW(),
        updated_at TIMESTAMP NOT NULL DEFAULT NOW(),
        FOREIGN KEY (category_id) REFERENCES material_category (id)
        CHECK (stock >= 0),
        CHECK (unit_cost >= 0)
    );

CREATE TABLE IF NOT EXISTS gift_material (
        id SERIAL PRIMARY KEY,
        material_quantity INTEGER NOT NULL DEFAULT 0,
        unit_of_measurement VARCHAR(64) NOT NULL,
        gift_id INTEGER NOT NULL,
        material_id INTEGER NOT NULL,
        FOREIGN KEY (gift_id) REFERENCES gift (id),
        FOREIGN KEY (material_id) REFERENCES material (id),
        CHECK (material_quantity > 0)
    );
```

3.0.2 Definizione delle operazioni

Operazione 1 Dati nome e cognome di un bambino, restituire i nomi e cognomi dei babbi delegati che gli hanno consegnato i regali.

```
SELECT DISTINCT deleg.id, deleg.name, deleg.surname
FROM delivery deliv, child c, delegate deleg
WHERE c.name = 'Mattia'
AND c.surname = 'Girolimetto'
AND c.id = deliv.child_id
AND deliv.delegate_id = deleg.id;
```

id	name	surname
1	John	Doe
3	Bob	Smith
2	Jane	Doe

Operazione 2 Restituire nome e cognome di tutti gli elfi che hanno prodotto almeno un regalo trasportato da un dato babbo delegato.

```
SELECT DISTINCT e.id, e.name, e.surname
FROM elf e
JOIN (
        SELECT g.elf_id
        FROM gift g
        JOIN (
                SELECT deliv.id
                FROM delivery deliv
                JOIN delegate deleg
                ON deleg.name = 'John'
                AND deleg.surname = 'Doe'
                AND deliv.delegate_id = deleg.id
        ) AS d
        ON d.id = g.delivery_id
) AS g
ON g.elf_id = e.id;
```

id	name	surname
1	Elfo	Doe
2	Elfa	Doe

Operazione 3 Restituire il costo totale per produrre un dato regalo.

```
SELECT SUM(m.unit_cost * gm.material_quantity)
AS Cost
FROM gift_material gm
JOIN material m
ON m.id = gm.material_id
WHERE gm.gift_id = 2;
```

cost
400

Operazione 4 Ricostruire l'albero genealogico di una data renna.

```
WITH RECURSIVE tmp (name, father_id, mother_id, father_name,
mother_name)
AS (
        -- Base case query
        SELECT r.name, r.father_id, r.mother_id, fr.name, mr.name
        FROM reindeer r
        -- Fetch father name from ID
        JOIN reindeer fr ON fr.id = r.father_id
        -- Fetch mother name from ID
        JOIN reindeer mr ON mr.id = r.mother_id
        WHERE r.name = 'Rudolph4'
        UNION ALL
        -- Recursive case query
        SELECT r2.name, r2.father_id, r2.mother_id, fr2.name,
 mr2.name
        FROM reindeer r2
        INNER JOIN tmp ON r2.id = tmp.father_id
    OR r2.id = tmp.mother_id
        -- Fetch father name
        JOIN reindeer fr2 ON fr2.id = r2.father_id
        -- Fetch mother name
        JOIN reindeer mr2 ON mr2.id = r2.mother_id
)
SELECT name, father_name, mother_name FROM tmp t;
```

name	father_name	mother_name
Rudolph4	Rudolph3	Blitzen2
Rudolph3	Comet	Donner2
Comet	Rudolph2	Blitzen
Rudolph2	Rudolph	Donner

Operazione 5 Restituire nomi e cognomi dei bambini presenti nella lista dei bambini correntemente buoni.

```
SELECT c.name, c.surname
FROM period_of_goodness pog
JOIN child c
ON c.id  = pog.child_id
WHERE pog.good_to IS NULL;
```

name	surname
Gabriele	Crestanello
Mattia	Girolimetto

Operazione 6 Trovare nomi e cognomi di tutti i bambini che in passato erano buoni e che correntemente non lo sono più.

```
SELECT c.name, c.surname
FROM period_of_goodness pog
JOIN child c
ON c.id = pog.child_id
WHERE NOT pog.good_to IS NULL
AND pog.child_id NOT IN (
        SELECT child_id
        FROM period_of_goodness pog2
        WHERE pog2.good_to IS NULL
);
```

name	surname
Grinch	Grinch

Operazione 7 Restituire lo stock rimanente dei materiali necessari per la costruzione dei regali destinati a bambini italiani.

```
SELECT m.name, m.stock
FROM material m
JOIN gift_material gm ON gm.material_id = m.id
JOIN gift g ON gm.gift_id = g.id
JOIN delivery d ON g.delivery_id = d.id
WHERE d.child_id IN (
        SELECT c.id
        FROM child c
        JOIN country co ON c.country_id = co.id
        WHERE co.name = 'Italy'
)
```

name	stock
Wood	10
Glue	15
Iron	20
Rubber	25

Operazione 8 Dato un elfo, restituire l'elenco di tutti i materiali (ordinati per categoria) che ha usato per la costruzione di regali nel corso del tempo.

```
SELECT m.name, mc.name AS category
FROM material m
JOIN material_category mc ON mc.id = m.category_id
JOIN gift_material gm ON gm.material_id = m.id
JOIN gift g ON gm.gift_id = g.id
WHERE g.elf_id IN(
        SELECT id
        FROM elf
        WHERE name = 'Elfo' AND surname = 'Doe'
)
ORDER BY mc.name
```

name	category
Glue	Elettronica
Wood	Legname

Operazione 9 Restituire nomi e cognomi dei bambini che hanno ricevuto una consegna da un determinato modello di slitta nel corso del tempo.

```
SELECT DISTINCT c.name, c.surname
FROM child c
JOIN delivery d ON d.child_id = c.id
JOIN delegate dg ON d.delegate_id = dg.id
WHERE dg.sled_id IN(
        SELECT s.id
        FROM sled s
        JOIN sled_model sm ON s.model_id = sm.id
        WHERE sm.manufacturer = 'Ferrari'
        AND sm.code = 'SF90'
)
```

name	surname
Mattia	Girolimetto

Operazione 10 Restituire nome, cognome e orario di lavoro di tutti gli elfi attualmente in attività.

```
SELECT e.name, e.surname, w.open_at AS opening,
        w.close_at AS closing
FROM elf e
JOIN workshop w ON e.workshop_id = w.id
WHERE e.working_to IS NULL
```

name	surname	opening	closing
Elfo	Doe	09:00:00	17:00:00
Elfa	Doe	08:00:00	16:00:00
Elfa	Johnson	08:00:00	16:00:00
Elfo	Williams	09:00:00	17:00:00

Operazione 11 Restituire i fornitori ordinati in base al valore dei materiali venduti e attualmente in magazzino.

```
SELECT s.company_name, SUM(m.unit_cost * m.stock) AS net
FROM supplier s
JOIN material_category mc ON mc.supplier_id = s.id
```

```
JOIN material m ON mc.id = m.category_id
GROUP BY s.id
ORDER BY net DESC
```

company_name	net
BigCo	8700
Acme Inc.	1800

Operazione 12 Data una renna restituire tutti i suoi fratelli e sorelle.

```
SELECT name, birth_date
FROM reindeer r
JOIN (
        SELECT r2.id, r2.father_id, r2.mother_id
        FROM reindeer r2
        WHERE r2.name = 'Cupid'
) AS j
ON j.mother_id = r.mother_id
AND j.father_id = r.father_id
AND j.id <> r.id
```

name	birth_date
Comet	2007-01-01

Operazione 13 Restituire la somma di tutti gli stipendi che babbo natale paga ai suoi dipendenti.

```
SELECT sum(d.wage) + SUM(e.wage) AS TOTAL
FROM delegate d, elf e
```

total
3.13698e+06

Capitolo 4

Testing

E' possibile visionare un'istanza del database e provare le query cliccando qui ed usando le seguenti credenziali:

- Sistema: PostgreSQL
- Server: db
- Utente: prof
- Password:jPasqxoWhmqk5PHp4AGmN1i6iLMrCyn2
- Database: babbodb