Progettazione logica e SQL DDL

corso di basi di dati e laboratorio

Prof. Alfio Ferrara

Anno Accademico 2017/2018

Indice

1 Progettazione logica

Fasi della progettazione logica

- **Ristrutturazione dello schema ER:** definizione di un nuovo schema ER ristrutturato eliminando opportunamente le gerarchie di generalizzazione e gli attributi composti e multivalore.
- **Traduzione dello schema ER:** definizione dello schema relazionale risultante dallo schema ER ristrutturato.

1.1 Ristrutturazione dello schema ER

Principi per la ristrutturazione dello schema ER

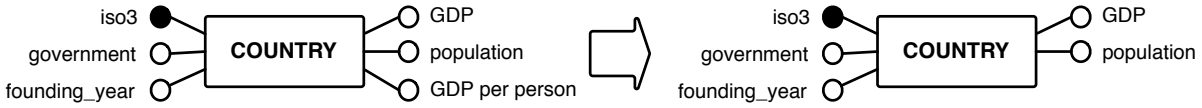
- Le modifiche apportate allo schema ER si basano su valutazioni legate ai parametri di carico della BD.
- Classi di modifiche:
- Analisi dei dati derivati (ridondanza)
- Eliminazione delle gerarchie di generalizzazione
- Scelta degli identificatori primari

Carico della base di dati

- Carico: insieme di applicazioni e operazioni che la BD dovrà gestire.
- Per valutare il carico operativo della BD è necessario conoscere informazioni circa:
- Volume dei dati (nr. medio di occorrenze di ogni entità/associazione, nr. medio di partecipazioni delle entità alle associazioni)
- Operazioni più frequenti
- Tipologia: interattiva o batch
- Frequenza: nr. medio di esecuzioni in un certo intervallo di tempo
- Dati coinvolti negli accessi: entità e/o associazioni

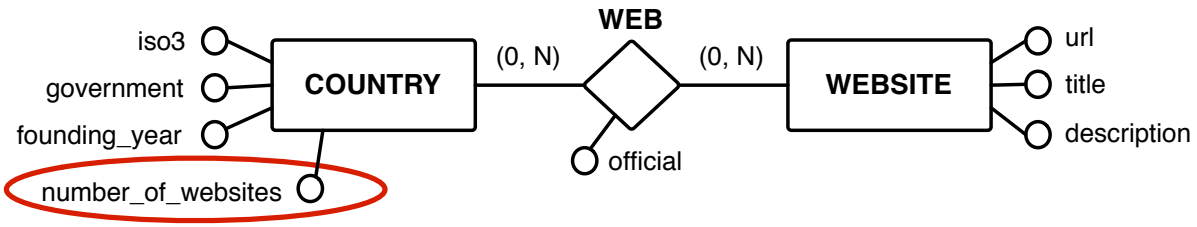
Dati derivati

- La presenza di attributi il cui valore è derivato (es., valore aggregato) può essere evitata con l'obiettivo di calcolare il valore derivato solo quando necessario mediante algoritmi a partire da altri valori memorizzati nella BD.
- **Vantaggi:** si elimina l'attributo e si riduce la quantità di dati memorizzati. Non c'è necessità di mantenere la coerenza fra dati primari e dati derivati.
- **Svantaggi:** maggiore overhead di elaborazione per calcolare il dato derivato.

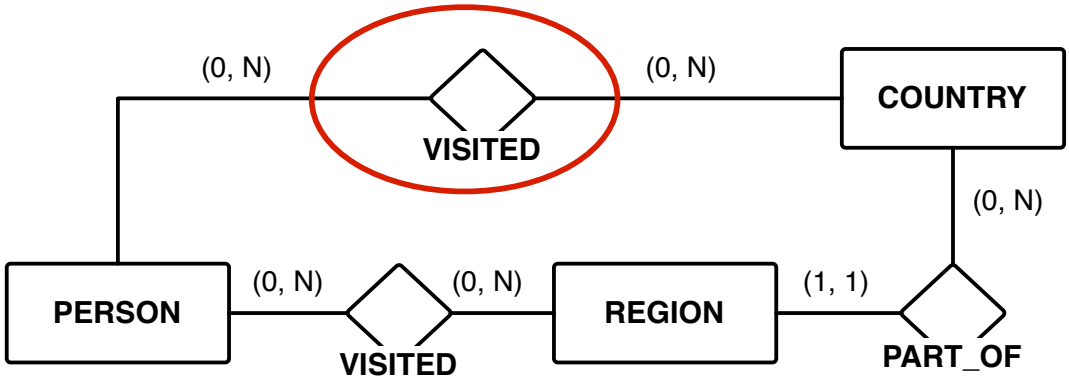


Il valore del prodotto interno lordo pro capite non necessita di essere memorizzato e può essere calcolato quando necessario utilizzando la formula: GDP / population

Valori di attributi derivabili da conteggio di occorrenze



Associazioni derivabili dalla composizione di altre associazioni



Analisi dei dati derivati

La convenienza nel mantenimento dei dati derivati va valutata (e documentata) tenendo conto di:

- In caso di eliminazione → Numero e frequenza delle operazioni necessarie a calcolare il dato derivato.
- In caso di mantenimento → Numero e frequenza delle operazioni di aggiornamento necessarie a mantenere la persistenza e coerenza dei dati derivati.
- Risparmio/uso della memoria secondaria.

Eliminazione delle gerarchie di generalizzazione

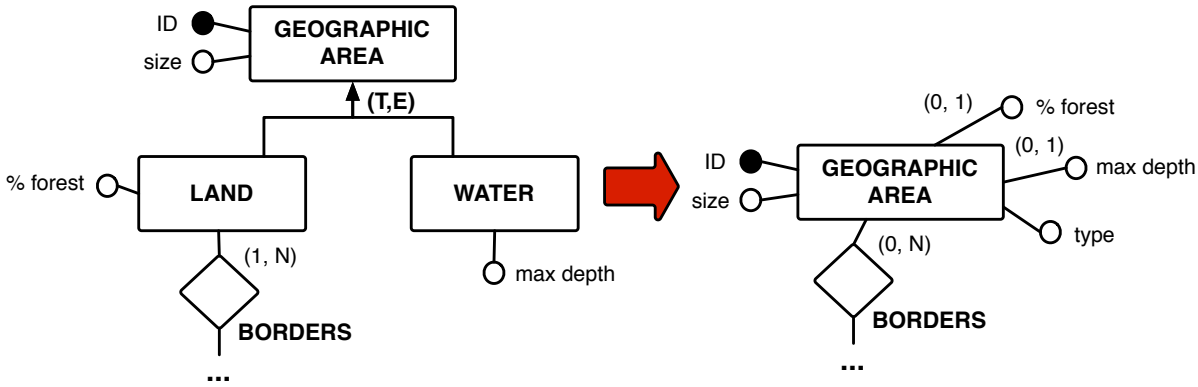
Alcuni modelli logici (fra cui il modello relazionale) non contemplano la nozione di gerarchia di generalizzazione. Pertanto occorre sostituire le gerarchie con entità e associazioni, rispettandone il più possibile la semantica.

1. Opzione 1: mantenimento della sola entità padre
2. Opzione 2: mantenimento delle sole entità figlie
3. Opzione 3: mantenimento di tutte le entità

Opzione 1: mantenimento della sola entità padre

- Aggiunta di un nuovo attributo *tipo* sull'entità padre per memorizzare la sottoclasse di ogni istanza (nel caso di gerarchie parziali, l'attributo tipo può assumere valori nulli).
- Aggiunta all'entità padre di tutti gli attributi e le associazioni propri delle entità figlie.
- Revisione delle cardinalità degli attributi aggiunti all'entità padre che assumono tutti cardinalità minima 0 (poiché potrebbero valere solo per alcune istanze).
- **Applicabilità: tutte le gerarchie.**

Opzione 1: esempio

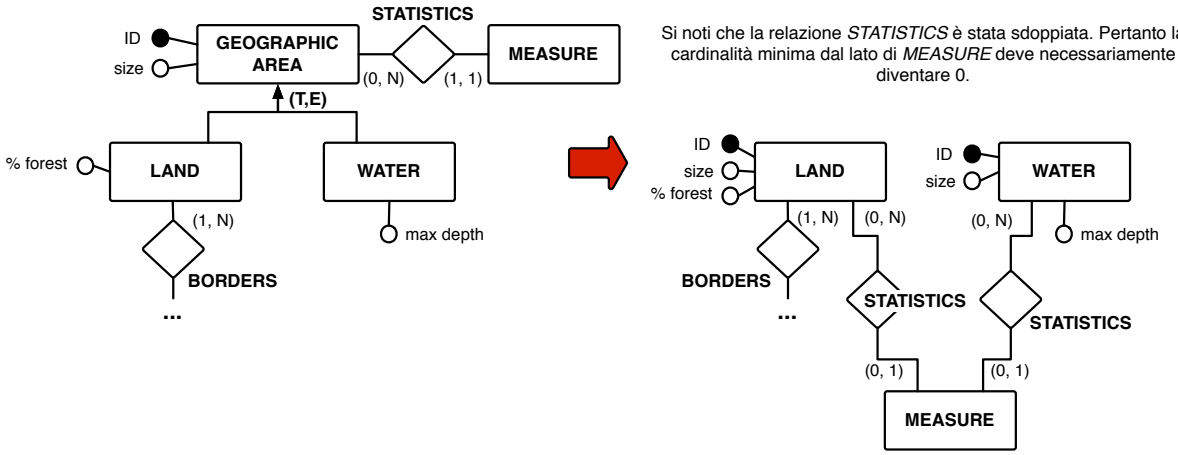


L'attributo *type* può avere solo due valori e è obbligatorio, poiché la gerarchia è totale (ovvero un'area geografica è necessariamente o una terra o uno specchio d'acqua). Se la gerarchia fosse parziale, l'attributo *type* avrebbe cardinalità minima 0. In caso di gerarchie sovrapposte, *type* diventa un attributo multivalore.

Opzione 2: mantenimento delle sole entità figlie

- Eliminazione dell'entità padre e trasferimento esplicito di tutti i suoi attributi e le sue relazioni su tutte le entità figlie.
- Non si alterano le cardinalità di tali attributi e relazioni.
- **Applicabilità: gerarchie totali e esclusive.**

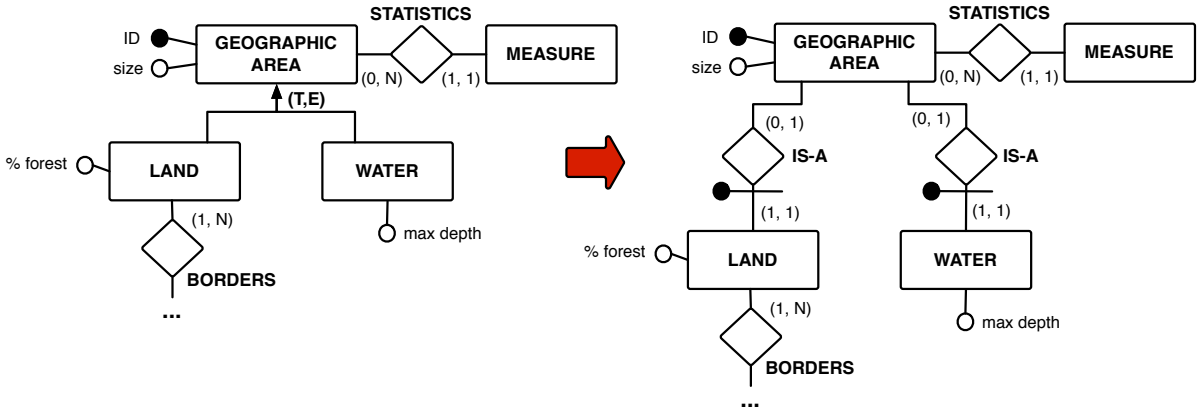
Opzione 2: esempio



Opzione 3: mantenimento di tutte le entità

- Si creano delle opportune associazioni 1:1 per rappresentare il legame *IS-A* espresso dalla gerarchia (ovvero il fatto che le entità figlie sono sottoinsiemi dell'entità padre e pertanto ogni loro istanza coincide (è) con una e una sola istanza dell'entità padre).
- Le entità figlie sono entità deboli rispetto all'entità padre, ovvero sono identificate dall'identificatore dell'entità padre.
- **Applicabilità: tutti i tipi di gerarchia.**

Opzione 2: esempio



Verifiche di correttezza

- Dopo gli aggiornamenti, occorre verificare che per ogni istanza delle specializzazioni esista un'istanza dell'entità generica. Per le **generalizzazioni totali**, occorre verificare che a ogni istanza della generalizzazione corrisponda un'istanza di qualche specializzazione.

Criteri di scelta della modalità di ristrutturazione

La maggioranza delle operazioni usa attributi di gerarchia, senza distinzione tra le istanze delle specializzazioni.

- **Opzione 1** $\rightarrow$ assicura un numero minore di accessi rispetto alle altre due alternative.

Criteri di scelta della modalità di ristrutturazione

La maggioranza delle operazioni usa contemporaneamente attributi dell'entità generica e attributi di singole specializzazioni e la gerarchia è di tipo (T,E).

- **Opzione 2** $\rightarrow$ assicura un risparmio di memoria rispetto all'opzione 1, per assenza di valori nulli. Riduce gli accessi rispetto all'opzione 3 perché non si visita l'entità generica per accedere a alcuni attributi.

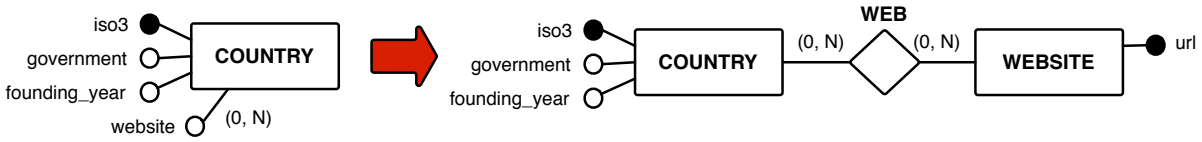
Criteri di scelta della modalità di ristrutturazione

La maggioranza delle operazioni usa attributi dell'entità generica e di sue specializzazioni, anche non contemporaneamente.

- **Opzione 3** $\rightarrow$ assicura un risparmio di memoria rispetto all'opzione 1, per assenza di valori nulli e riduce i tempi di accesso rispetto all'opzione 2 per un numero minore di attributi.

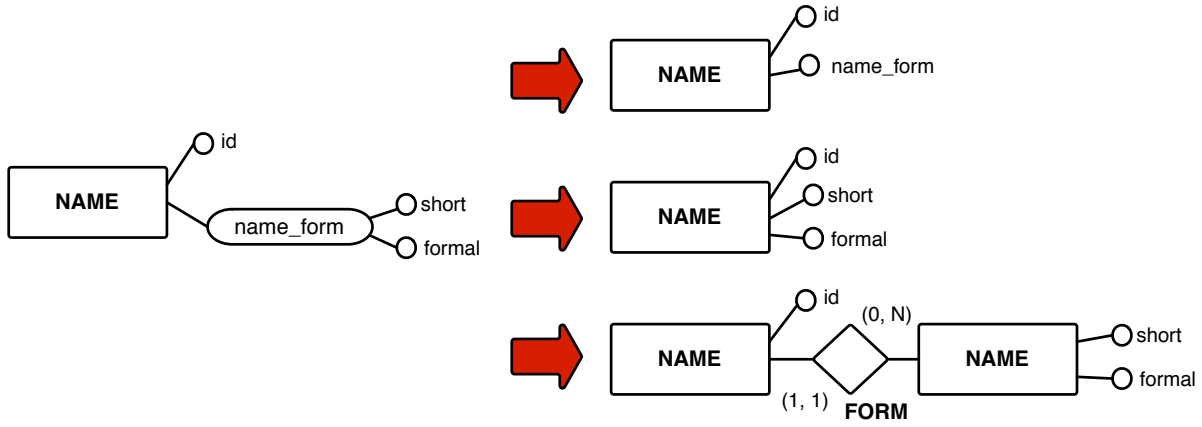
Eliminazione di attributi multivalore

- Nel caso di attributi multivalore di un'entità E, si procede a *reificare* l'attributo in un'entità E_a e si introduce una nuova relazione R fra E e E_a, la cui cardinalità è uguale alla cardinalità dell'attributo multivalore.

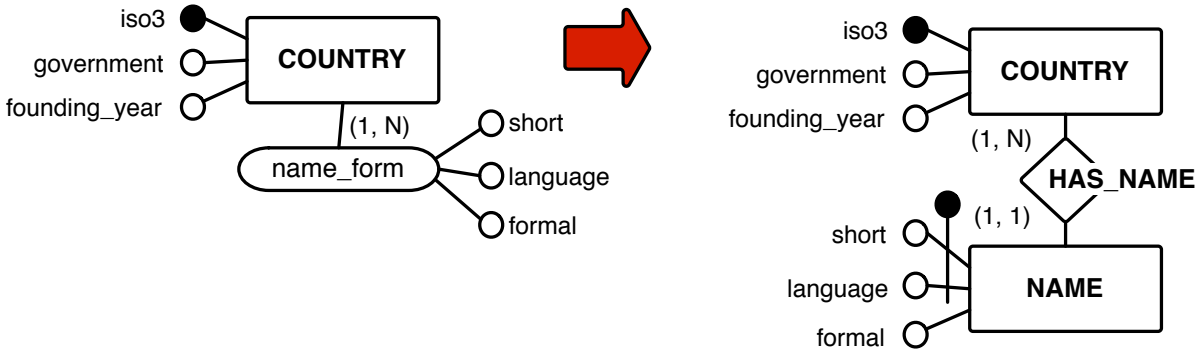


Eliminazione di attributi composti

- Nel caso di attributi composti e **non multivalore** di un'entità E, si procede è possibile procedere semplicemente memorizzando le componenti dell'attributo come un unico attributo o come singoli attributi separati.
- Per evitare problemi dovuti a eventuali dipendenze transitive è possibile anche creare una nuova entità e collegarla da una relazione con cardinalità 1:N.



Esempio con attributi composti e multivalore



1.2 Traduzione dello schema ER

Metodologia generale

La traduzione dallo schema ER ristrutturato allo schema relazionale può essere vista come un processo composto principalmente da tre fasi:

1. Traduzione di entità
2. Traduzione di relazioni
3. Verifica di normalizzazione

Traduzione di entità

Data un'entità

$$E = \{K, W\}$$

con:

$$K = \{a_1, a_2, \ldots, a_t\} \text{ come identificatore e}$$

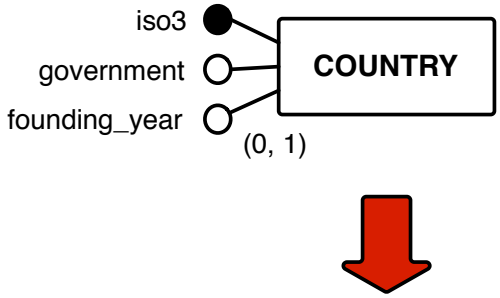
$$W = \{a_{(t+1)}, a_{(t+2)}, \ldots, a_{(t+n)}\} \text{ come insieme di attributi descrittivi}$$

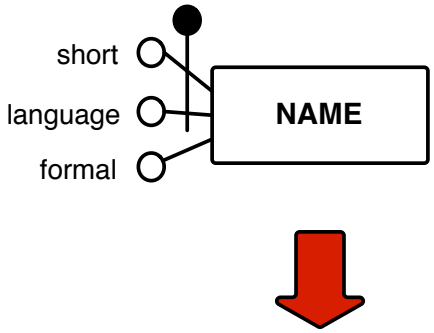
E è tradotto un una relazione

$$E(\underline{a_1, a_2, \ldots, a_t}, a_{(t+1)}, a_{(t+2)}, \ldots, a_{(t+n)})$$

in cui K è la chiave primaria.

Esempi





COUNTRY(iso3, government, founding_year*)

NAME(short, language, formal)

SI noti che eventuali attributi non obbligatori (i.e., con cardinalità minima = 0) potranno assumere valore nullo nelle corrispondenti relazioni del modello relazionale (es. founding_year)

Traduzione di associazioni

La traduzione di associazioni dal modello ER ristrutturato al modello relazionale permette diverse opzioni che dipendono:

- Dalla tipologia dell'associazione (i.e., binaria o n-aria)
- Dalla tipologia di cardinalità (1:1, 1:N, N:M)

Associazioni molti-a-molti (N:M)

Data un'associazione $R = \{C\}$ fra le entità $E1 = \{K1, W1\}$ e $E2 = \{K2, W2\}$ in cui:

- max-card$(E1, R) = N$, max-card$(E2, R) = N$
- C insieme di (eventuali) attributi della relazione R
- $K1$ e $K2$ identificatori di $E1$ e $E2$, rispettivamente
- $W1$ e $W2$ attributi descrittivi di $E1$ e $E2$, rispettivamente

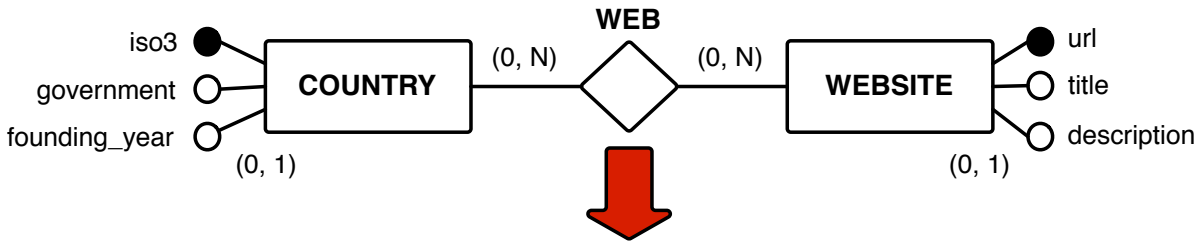
si traduce R con una relazione:

$$R(\underline{K1, K2}, C)$$

In cui C è presente solo nel caso in cui l'associazione R abbia attributi propri. La chiave primaria è composta da $K1 \cup K2$. Solo nel caso in cui si vogliano rendere possibili più relazioni fra le stesse istanze di $E1$ e $E2$ al variare di C, si include anche C nella chiave primaria. Ciò avviene ad

esempio nelle serie storiche, in cui tipicamente C è una data e è possibile avere più relazioni fra le stesse istanze di $E1$ e $E2$ in momenti diversi nel tempo.

Esempio 1



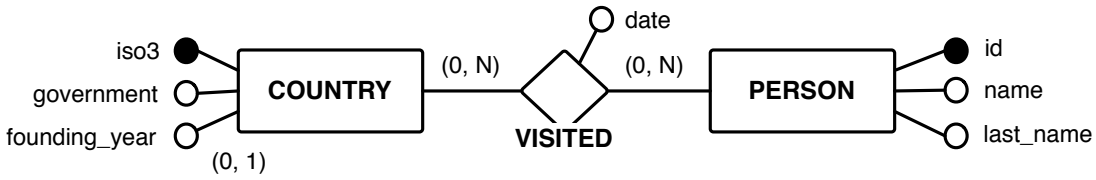
COUNTRY(<u>iso3</u>, government, founding_year*)

WEBSITE(<u>url</u>, title, description*)

WEB(<u>*iso3, url*</u>)

Gli attributi iso3 *e* url *della relazione WEB sono chiavi esterne. Le chiavi esterne possono avere anche altri nomi rispetto agli attributi a cui fanno riferimento. Spesso si usa come nomi i nomi delle entità a cui si riferiscono, ovvero:*
WEB(<u>*country, website*</u>)

Esempio 2



COUNTRY(<u>iso3</u>, government, founding_year*)

PERSON(<u>id</u>, name, last_name)

VISITED(<u>*country, person*</u>, date)

Soluzione 1: *l'attributo* date *non è parte della chiave, perciò possiamo registrare solo una visita per persona e country (non conservando lo storico dei viaggi).*

COUNTRY(<u>iso3</u>, government, founding_year*)

PERSON(<u>id</u>, name, last_name)

VISITED(<u>*country, person*</u>, <u>date</u>)

Soluzione 2 (serie storica): *l'attributo* date *è parte della chiave, perciò possiamo registrare molte visite per persona e country purché in date diverse (conservando lo storico dei viaggi).*

Associazioni 1:1

Data un'associazione $R = \{C\}$ fra le entità $E1 = \{K1, W1\}$ e $E2 = \{K2, W2\}$ in cui:

- max-card$(E1, R) = 1$, max-card$(E2, R) = 1$
- C insieme di (eventuali) attributi della relazione R
- $K1$ e $K2$ identificatori di $E1$ e $E2$, rispettivamente
- $W1$ e $W2$ attributi descrittivi di $E1$ e $E2$, rispettivamente

si traduce la relazione R con una chiave esterna inserita in $E1$ o $E2$:

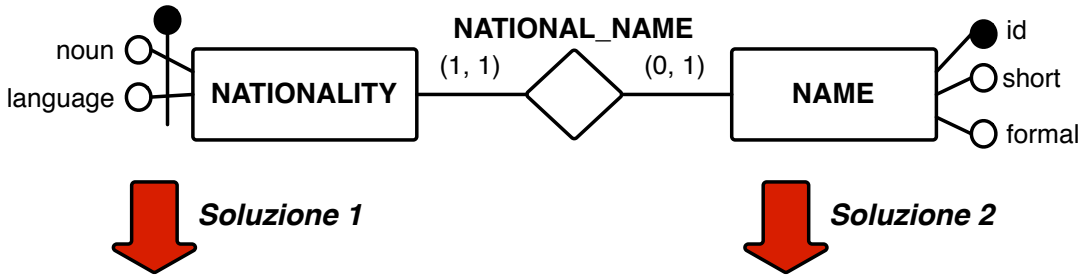
$$E1(K1, W1, K2, C)E2(K2, W2)$$

oppure

$$E1(K1, W1)E2(K2, W2, K1, C)$$

si noti che gli eventuali attributi C dell'associazione "seguono" la chiave esterna. La chiave esterna ammette valori nulli se la cardinalità minima dell'associazione è 0. Nel caso di entità deboli, la chiave esterna è anche la chiave primaria.

Esempio



NATIONALITY(<u>noun, language</u>, *name*)
NAME(<u>id</u>, short, formal)

NATIONALITY(<u>noun, language</u>)
NAME(<u>id</u>, short, formal, *noun*, language**)

Associazioni 1:N

Data un'associazione $R = \{C\}$ fra le entità $E1 = \{K1, W1\}$ e $E2 = \{K2, W2\}$ in cui:

- max-card$(E1, R) = 1$, max-card$(E2, R) = N$ (o viceversa)
- C insieme di (eventuali) attributi della relazione R

- $K1$ e $K2$ identificatori di $E1$ e $E2$, rispettivamente
- $W1$ e $W2$ attributi descrittivi di $E1$ e $E2$, rispettivamente

si traduce la relazione R con una chiave esterna inserita nella relazione corrispondente all'entità con cardinalità massima 1:

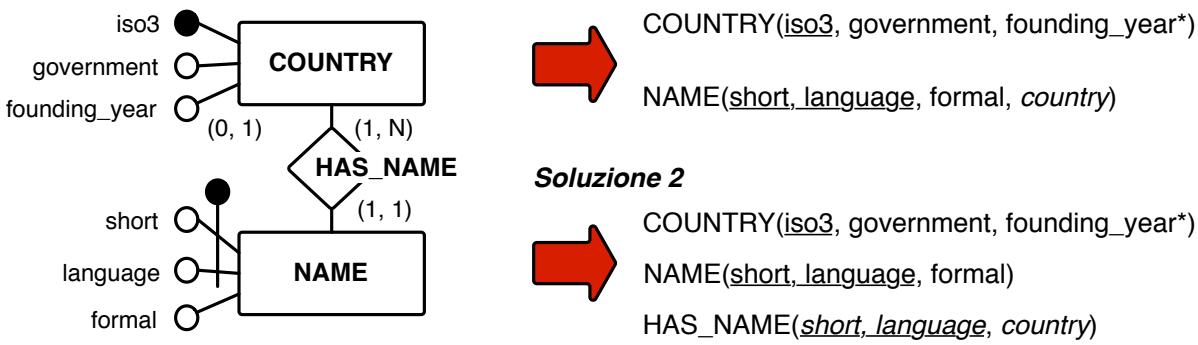
$$E1(K1, W1, K2, C)E2(K2, W2)$$

oppure si può creare una nuova relazione in cui però la chiave è composta dalla sola chiave esterna

$$R(\underline{K1}, W1, K2, C)$$

si noti che gli eventuali attributi C dell'associazione "seguono" la chiave esterna. La chiave esterna ammette valori nulli se la cardinalità minima dell'associazione è 0.

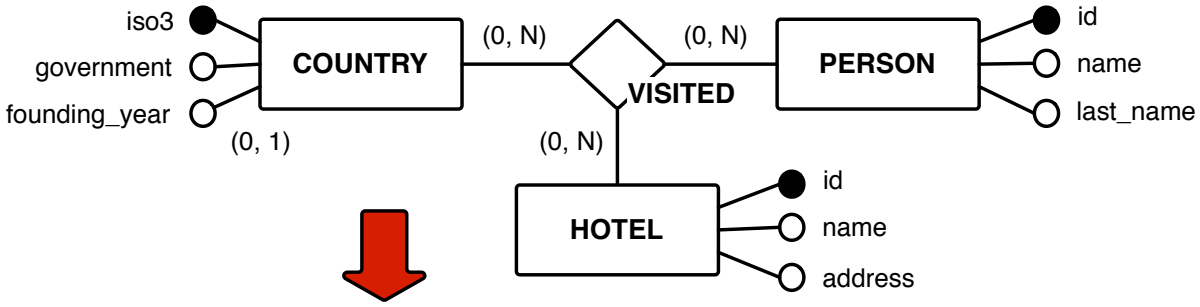
Esempio



Altre tipologie di associazione

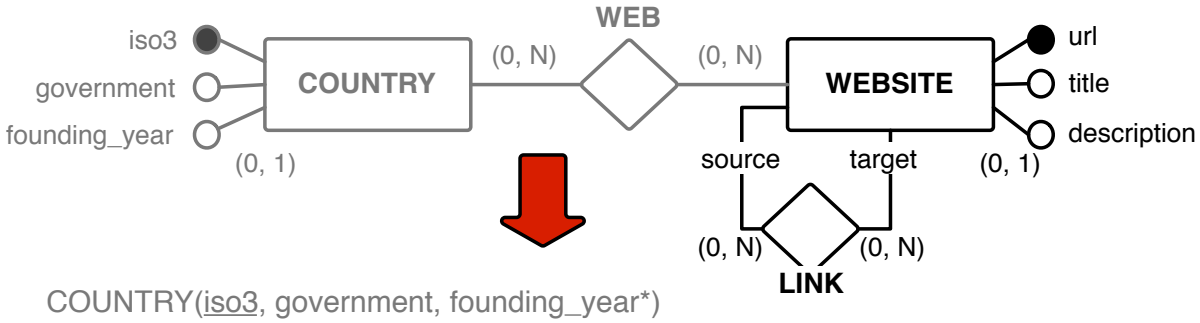
- **Associazioni n-arie:** si traducono come le relazioni N:M con una chiave esterna per ogni entità implicata nell'associazione.
- **Associazioni ricorsive:** si traducono come normali associazioni ma ridenominando le chiavi esterne in modo da evidenziarne il ruolo nell'associazione.

Esempio di relazione n-aria



COUNTRY(iso3, government, founding_year*)
PERSON(id, name, last_name)
HOTEL(id, name, address)
VISITED(*country, person, hotel*, date)

Esempio di relazione ricorsiva



COUNTRY(iso3, government, founding_year*)
WEB(*iso3, url*)
WEBSITE(url, title, description*)
LINK(*source, target*)

1.3 Reverse engineering

Reverse engineering

- Per reverse engineering si intende l'operazione di derivazione dello schema ER a partire dallo schema relazionale.
- E' utile quando occorre intervenire su un database pre-esistente e non documentato.
- Il criterio principale è di partire dalle chiavi esterne per ricavare le associazioni e le cardinalità.
- Nello schema ER **non** ci sono chiavi esterne.

2 Creazione dello schema relazionale con SQL DDL

2.1 Introduzione a SQL

SQL, Structured Query Language

- **SEQUEL:**

  - Structured English QUEry Language

- **'70-'80**

  - Linguaggio sviluppato per il DBMS relazionale System R (IBM San Jose, CA)

- **'86**

  - Primo standard di SQL (ANSI)

  - Buone funzionalità di DML

  - Limitate funzionalità DDL

SQL, Structured Query Language

- **'89**

  - Estensione dello standard (integrità referenziale) **SQL-89**

- **'92**

  - Seconda versione dello standard (aggiunta di molte funzionalità DDL) **SQL-92** o **SQL-2**

- **Oggi**

  Nuova versione dello standard con una serie di estensioni (e.g., trigger, tipi composti, viste ricorsive, supporto per rappresentazione di oggetti di grandi dimensioni) **SQL-99** o **SQL-3**

2.2 Domini

Domini elementari

- Domini numerici.
- Domini di tipo carattere.
- Domini temporali.
- I tipi elementari sono ulteriormente suddivisi in vari sottotipi.

Domini numerici

- Tipi numerici esatti: rappresentano valori interi e valori decimali in virgola fissa.

  `INTEGER, SMALLINT, NUMERIC, DECIMAL`

- Tipi numerici approssimati: rappresentano valori numerici approssimati con rappresentazione in virgola mobile.

  `REAL, DOUBLE, FLOAT`

Stringhe e caratteri

- `CHARACTER [VARYING]`

  Questo dominio (spesso abbreviato in `CHAR [VARCHAR]`) permette di rappresentare singoli caratteri o stringhe, di lunghezza fissa o variabile e definire stringhe di caratteri di lunghezza massima predefinita.

- `CHARACTER [VARYING] [(n)]` indica stringhe di lunghezza [massima] n.

Altri tipi di stringa

- **BIT**

  Questo dominio permette di rappresentare stringhe di bit, dove ciascun elemento della stringa può assumere solo il valore 0 o il valore 1.

  `BIT [VARYING] [(n)]` (vedi `CHAR`)

- **BOOLEAN**

  Dominio che permette di rappresentare singoli valori booleani `TRUE` e `FALSE` (restrizione di BIT).

Domini temporali

- **DATE**

  Rappresenta le date nel formato `YYYY-MM-DD` con campi *year* (4 cifre), *month* (2 cifre comprese tra 1 e 12) e *day* (2 cifre comprese tra 1 e 31 ed ulteriori restrizioni a seconda del mese).

- **TIME**

  Rappresenta i tempi `HH:MM:SS` con campi *hour* (2 cifre), *minute* (2 cifre) e *second* (2 cifre).

- **TIMESTAMP**

  Comprende entrambi `DATE` e `TIME`.

Altri domini

- **BLOB e CLOB**

  Domini che permettono di rappresentare oggetti di grandi dimensioni, a supporto di gestione di dati multimediali e semistrutturati del sistema informativo.

- **BLOB, binary large object** sequenza di valori binari.
- **CLOB, character large object** sequenza di caratteri.
- Il sistema garantisce la memorizzazione, ma non è possibile utilizzare il valore come criterio di selezione per interrogazioni.

2.3 Creazione dello schema

Creazione dello schema

`CREATESCHEMA` NomeSchema
`AUTHORIZATION` IDAutorizzazione {ElementoSchema}

IDAutorizzazione = utente/account proprietario dello schema (se omesso, è l'utente che ha lanciato il comando).

NomeSchema = se omesso, coincide con il nome del proprietario.

Elementi dello schema: domini, tabelle, asserzioni, viste, privilegi.

Definizione di tabelle

`CREATE TABLE` NomeTabella
(NomeAttributo Dominio [ValoreDefault] [Vincoli]
{, NomeAttributo Dominio [ValoreDefault] [Vincoli]}
AltriVincoli)

Esempio

COUNTRY(iso3, government, founding_year, longitude, latitude)

```
CREATE TABLE country(
   iso3 CHAR(3) PRIMARY KEY,
```

```
    government VARCHAR(20),
    founding_year INTEGER(4),
    longitude DOUBLE PRECISION,
    latitude DOUBLE PRECISION
);
```

Valori di default

Valore che deve assumere l'attributo quando viene inserita una tupla nella tabella senza che sia specificato un valore per l'attributo stesso.

`DEFAULT ⟨GenericoValore | User | NULL⟩`

GenericoValore = valore compatibile con il dominio. Risultato della valutazione di un'espressione o definito come valore costante.

User = come valore di default si considera l'identificativo dell'utente che esegue il comando di aggiornamento.

NULL = corrisponde al valore di default di base.

Il valore predefinito di default è NULL per gli attributi sui quali non è stato definito un vincolo `NOT NULL`.

Es. `durata integer DEFAULT 120.`

2.4 Vincoli

Vincoli intrarelazionali

Vincolo: proprietà che deve essere verificata da ogni istanza della BD.

NOT NULL: il valore nullo non è ammesso come valore dell'attributo.

Es. `government char(20) NOT NULL`

UNIQUE: righe diverse non possono possedere gli stessi valori in corrispondenza di un attributo o di un insieme di attributi. Il valore nullo può comparire su diverse righe.

Es. `UNIQUE(longitude, latitude)`

Esempio

```
CREATE TABLE country(
  iso3 CHAR(3) PRIMARY KEY,
  government VARCHAR(20),
  founding_year INTEGER(4) NOT NULL,
  longitude DOUBLE PRECISION,
```

```
  latitude DOUBLE PRECISION,
  UNIQUE(longitude, latitude)
);
```

Chiave primaria

PRIMARY KEY

Specifica della chiave primaria. Una sola volta per ogni tabella; **direttamente su un attributo** oppure **specificando gli attributi** che costituiscono la chiave primaria.

Esempio:

`iso3 PRIMARY KEY`

oppure

`PRIMARY KEY(iso3)` (usato al termine della lista di attributi)

Se la chiave è composta da più di un attributo, la seconda soluzione è obbligatoria.

Esempio:

NAME(short, language, formal)

`short PRIMARY KEY`

`language PRIMARY KEY`

è sbagliato! Per le chiavi composte occorre usare

`PRIMARY KEY(short, language)`

Vincoli inter-relazionali

Integrità referenziale (foreign key)

Legame tra i valori dell'attributo (o attributi) della tabella corrente (**che riferisce** o **referente**) e i valori dell'attributo (o attributi) di un'altra tabella (**riferita**). Impone che per ogni tupla della tabella che riferisce, il valore dell'attributo, se diverso da nullo, sia presente tra i valori di un attributo delle tuple della tabella riferita.

Dichiarazione del vincolo

Costrutto `REFERENCE`

Costrutto `FOREIGN KEY` (obbligatorio se la chiave esterna è composta da più attributi).

Esempio:

COUNTRY(iso3, government*, founding_year)

NAME(short, language, formal, country)

```
CREATE TABLE name (
  short VARCHAR(10),
```

```
  language VARCHAR(10),
  PRIMARY KEY(short, language),
  country CHAR(3) NOT NULL REFERENCES country(iso3),
);
```

Dichiarazione del vincolo

Esempio in cui l'uso del costrutto foreign key è obbligatorio poiché la chiave esterna è composta:

COUNTRY(iso3, government*, founding_year)
NAME(short, language, formal)
HAS_NAME(short, language, country)

```
CREATE TABLE has_name(
  short VARCHAR(10),
  language VARCHAR(10),
  PRIMARY KEY(short, language),
  country CHAR(3) NOT NULL REFERENCES country(iso3),
  FOREIGN KEY(short, language) REFERENCES name(short, language)
);
```

Integrità referenziale

Possibilità di associare una *azione referenziale innescata* alle violazioni del vincolo di integrità referenziale generate da operazioni di modifica sulla tabella riferita, ovvero:

- cancellazione di tuple riferite da chiave esterna;
- modifiche del valore dell'attributo chiave riferito da chiave esterna.

Si tratta di azioni che il DBMS esegue in caso di cancellazioni/aggiornamenti sulla tabella riferita.

Azioni

CASCADE: i valori di chiave esterna della tabella che riferisce (nel nostro esempio `has_name`) corrispondenti alla tupla modificata/cancellata nella tabella riferita (nel nostro esempio `country` e `name`) vengono a loro volta modificati/cancellati.

SET NULL: i valori di chiave esterna della tabella che riferisce corrispondenti alla tupla modificata/cancellata nella tabella riferita sono posti a NULL.

SET DEFAULT: i valori di chiave esterna della tabella che riferisce corrispondenti alla tupla modificata/cancellata nella tabella riferita sono posti al valore predefinito.

NO ACTION: rifiuta l'azione di aggiornamento/cancellazione sulla tabella riferita se vi sono tuple che fanno riferimento alla tupla da modificare/cancellare nella tabella che riferisce.

Esempio

```
CREATE TABLE has_name(
  short VARCHAR(10),
  language VARCHAR(10),
  PRIMARY KEY(short, language),
  country CHAR(3) NOT NULL REFERENCES country(iso3),
   ON UPDATE CASCADE ON DELETE NO ACTION
  FOREIGN KEY(short, language) REFERENCES name(short, language)
   ON UPDATE CASCADE ON DELETE CASCADE
);
```

Vincoli di dominio e di ennupla

- Vincoli di integrità che impongono condizioni generiche sui valori delle tuple in una tabella (e.g., vincoli dipendenti dall'applicazione).
- I vincoli sono specificati tramite l'uso di una o più clausole `CHECK` nel comando per la definizione di tabelle:

  `CHECK` ⟨check-condition⟩

- La condizione di vincolo (check-condition) è un qualsiasi predicato (o combinazione booleana di predicati) che può apparire nella clausola `WHERE` di una interrogazione SQL.

Esempio.

```
CREATE TABLE country
   iso3 CHAR(3) PRIMARY KEY,
   government VARCHAR(20),
   founding_year INTEGER(4) NOT NULL,
   longitude DOUBLE PRECISION,
   latitude DOUBLE PRECISION
    CHECK (founding_year < 2015)
    CHECK (longitude <= -180 AND longitude >= 180)
    CHECK (latitude <= -90 AND latitude >= 90)
    CHECK ((government IN ('republic', 'monarchy',
     'constitutional monarchy', ...))
```

Definizione di domini

```
CREATE DOMAIN NomeDominio AS
TipoDatoBase [ ValoreDefault ] {Vincoli }
```

Vincoli: esprime condizioni sui valori ammessi dal dominio, secondo la sintassi prevista per la definizione dei vincoli di integrità.

Esempio:

```
CREATE DOMAIN government AS VARCHAR(20)
DEFAULT 'republic'
CHECK (VALUE IN ('republic', 'monarchy', 'constitutional monarchy',
...))
```

2.5 Modifica dello schema

Comandi di modifica dello schema: ALTER

```
ALTER TABLE ⟨ NomeTabella
  ALTER COLUMN NomeAttributo
   ⟨SET DEFAULT NuovoDefault
    | DROP DEFAULT ⟩|
  ADD CONSTRAINT DefVincolo |
  DROP CONSTRAINT NomeVincolo |
  ADD COLUMN DefAttributo |
```

```
  DROP COLUMN NomeAttributo ⟩
```

Esempio:

```
ALTER TABLE country ADD COLUMN president VARCHAR(20)
```

Comandi di modifica dello schema: DROP

```
DROP ⟨ SCHEMA | DOMAIN | TABLE | VIEW | ASSERTION ⟩
  NomeElemento [RESTRICT | CASCADE]
```

- Opzione `RESTRICT` (default):

  Il comando non è eseguito in presenza di elementi non vuoti (es., uno schema non è rimosso se contiene tabelle o altri elementi; una tabella non è rimossa se possiede delle righe o se è riferita in qualche altra tabella).

- Opzione `CASCADE`:

  Tutti gli elementi specificati devono essere rimossi (eliminare un elemento che ne contiene altri implica la rimozione anche di questi ultimi).

Catalogo relazionale

- I comandi DDL si traducono in operazioni sul catalogo della base di dati.
- Catalogo relazionale: descrizione delle tabelle che costituiscono la base di dati e dei vari elementi dello schema.
- Nei DBMS relazionali il catalogo è costituito a sua volta da tabelle (riflessività).