Regole di conversione dal modello concettuale al modello logico relazionale

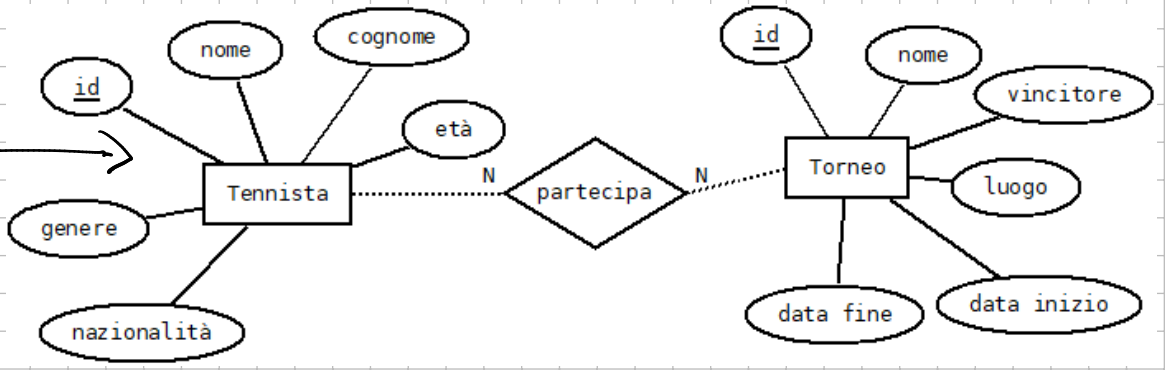
Modello ER $\xrightarrow{\text{regole}}$ Modello logico

- Ogni entità del diagramma ER diventa una tabella del modello logico
- Gli attributi di un'entità diventano le intestazioni di colonna della tabella relativa
- Per gli attributi vanno specificati i tipi, senza entrare nel dettaglio
- Per le relazioni si hanno 3 casi, dipende dalla cardinalità
  - N:N : la relazione diventa una tabella, il cui nome va eventualmente adattato. Questa tabella conterrà i propri eventuali attributi più i due id delle entità che sono in relazione

i nomi degli id saranno id_nome_entità (per convenzione) e sono chiamati chiavi esterne (foreign key)

- 1:N: la chiave primaria dell'entità che ha vicino l'1 diventa chiave esterna dell'entità che ha vicino la N

Tennista (id : intero, nome : stringa, cognome : stringa,
età : intero, genere : enum, nazionalità : stringa)



Partecipa (id_tennista : intero, id_torneo : intero
id : intero)

Tennista

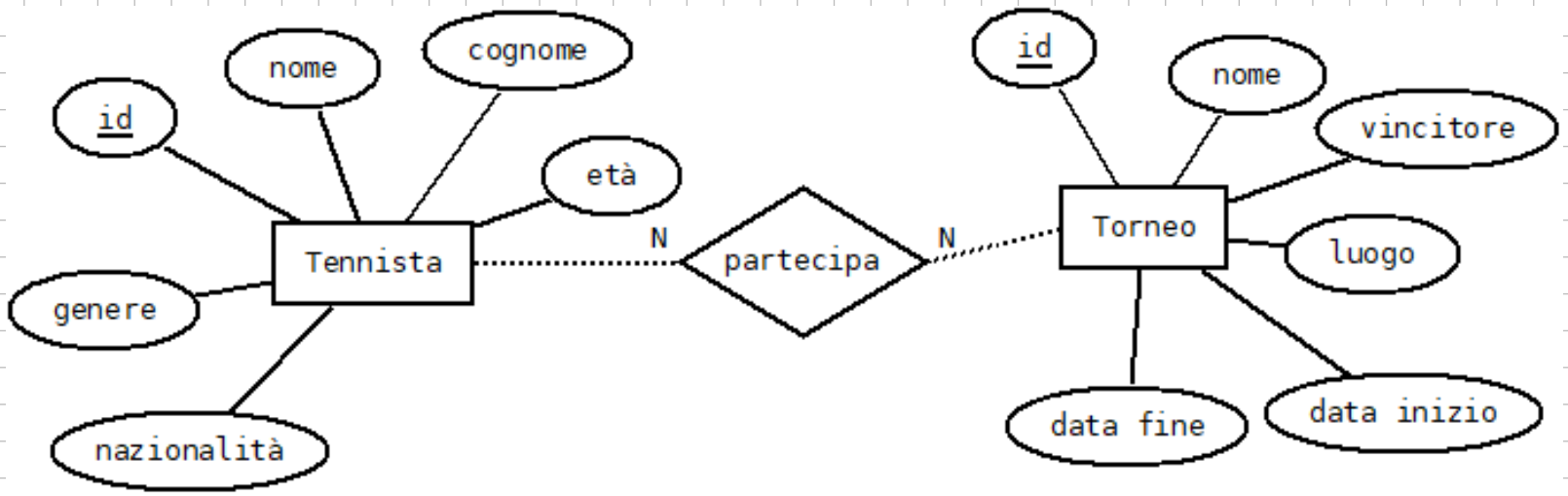
id	nome
1	Piero
2	Anna
3	Gianni

Torneo

id	nome
1	Milano
2	Genova

Partecipa

id	id_tennista	id_torneo
1	2	1
2	1	1



Tennista (id : intero ; nome : stringa ; cognome : stringa ; età : intero ; genere : enum ; nazionalità : stringa)

Torneo (id : intero ; nome : stringa ; vincitore : stringa ; luogo : stringa ; data_inizio : data ; data_fine : data)

Partecipa (id_tennista : intero ; id_torneo : intero ; id : intero)

```
CREATE TABLE tennista(
    id int unsigned auto_increment primary key,
    nome varchar(50) not null,
    cognome varchar(50) not null,
    data_nascita date not null,
    genere enum('M', 'F', 'NA'),
    nazionalità varchar(50)
);
CREATE TABLE partecipa(
    id int unsigned auto_increment primary key,
    id_tennista int unsigned NOT NULL,
    id_torneo int unsigned NOT NULL
)
```

```
CREATE TABLE torneo(

)
```

Aggiunta dei vincoli di integrità referenziale

```
CREATE TABLE partecipa(
  id int unsigned auto_increment primary key,
  id_tennista int unsigned NOT NULL,
  id_torneo int unsigned NOT NULL,
  CONSTRAINT FK_id_tennista
   FOREIGN KEY id_tennista
   REFERENCES tennista(id)
   ON UPDATE CASCADE
   ON DELETE CASCADE
```

Aggiunta delle chiavi uniche

UNIQUE

Esempio con il nome del torneo

nome varchar(100) NOT NULL UNIQUE,

Esempio con nome e data_inizio

In questo caso il vincolo viene applicato dopo

UNIQUE (nome, data_inizio)

- Aggiunta di vincoli di dominio

Esempio di vincolo sul valore

```
voto int unsigned NOT NULL check(voto <= 10
                                 AND voto >= 0)
```

Esempio di vincolo su più valori:

```
data_fine date check (data_fine > data_inizio)
```

Aggiunta di indici

```
CREATE INDEX index-cognome
ON Tennista (cognome)
```

migliora le performance di operazioni che coinvolgono quell'attributo

Istruzioni per la modifica di una tabella

Aggiunta di una colonna

```
ALTER TABLE nome_tabella
ADD COLUMN nuovo_attributo tipo
```

Modifica di una colonna (tipo)

```
ALTER TABLE nome_tabella
MODIFY COLUMN nome_attributo nuovo_tipo
```

Modifica di una colonna (nome)

ALTER TABLE nome_tabella

CHANGE COLUMN nome nuovo_nome tipo

Eliminazione di una colonna

ALTER TABLE nome_tabella

DROP COLUMN nome_attributo

Aggiunta di un vincolo di chiave esterna

```
ALTER TABLE nome_tabella
ADD CONSTRAINT nome_chiave
    FOREIGN KEY (chiave_esterna)
    REFERENCES tabella(chiave_primaria)
```

Rimozione di un vincolo di chiave esterna

```
ALTER TABLE nome_tabella
DROP FOREIGN KEY nome_chiave
```

Rinominare una tabella

ALTER TABLE nome_tabella
RENAME TO nuovo_nome

Rimuovere una tabella

DROP TABLE nome_tabella

Non è più recuperabile

Istruzioni DCL (Data Control Language)

I DBMS sono software che supportano la multiutenza, quindi sono nativamente pensati per gestire autenticazione e autorizzazione

In MariaDB esiste un utente principale, root, che è autorizzato a fare tutto.

Non è una buona idea utilizzarlo nelle applicazioni, serve per creare schemi e utenti

L'utente deve avere i minimi privilegi possibili per fare quello che deve fare

Creare un nuovo utente

CREATE USER nome@dove
IDENTIFIED BY password

Assegnare privilegi

GRANT privilegi ON database TO utente

GRANT ALL PRIVILEGES ON database.*
TO utente

Cambiare la password di un utente

ALTER USER user@dove IDENTIFIED BY nuova_password

Cancellare un utente

DROP USER nome_utente

Istruzioni per la manipolazione dei dati: DML (Data Manipulation Language)

CREATE → INSERT Inserisce righe

READ / RETRIEVE → SELECT Estrae dati

Update → UPDATE Aggiorna righe

DELETE → DELETE Cancella righe

Inserimento dei dati

INSERT INTO nome_tabella
(elenco_attributi) VALUES(elenco_valori)

Ci deve essere corrispondenza tra
l'elenco dei valori e quelli degli attributi,
sia in quantità che posizione

Modifica dei dati

UPDATE nome_tabella
SET elenco coppie attributo = nuovo_valore
[WHERE condizione]

La clausola WHERE è opzionale, ma quasi sempre utilizzare per individuare la riga/righe da modificare.
Se non è presente la modifica viene applicata a tutte le righe

Eliminazione dei dati

DELETE FROM nome_tabella
[WHERE condizione]

Attenzione: l'operazione è irreversibile
e se non ci fosse la clausola WHERE
cancellerebbe tutte le righe

Interrogazione dei dati

QL (Query Language)

Il minimo indispensabile

```
SELECT *
FROM nome-tabella
[WHERE condizione]
```