Database relazionali (introduzione)

Liceo G.B. Brocchi – Bassano del Grappa (VI)
Liceo Scientifico – opzione scienze applicate
Giovanni Mazzocchin

Database - introduzione

- Ciascuno di noi interagisce quotidianamente con i database: operazioni quali *visualizzare il proprio estratto conto tramite home banking*, *prenotare un biglietto aereo*, *fare acquisti online* richiedono sicuramente degli accessi ad uno o più database
- L'evoluzione dei database ha costituito una parte fondamentale nel progresso dell'Informatica: immaginate quanto potesse essere difficile e scomodo gestire anche solo i dati anagrafici di un comune in forma cartacea…
- Gli esempi elencati sopra possono essere considerati **applicazioni tradizionali** dei database, in quanto l'informazione memorizzata nei database è prevalentemente testuale o numerica
- Ci sono altri esempi di applicazioni più moderne dei database: **big data**, **sistemi informativi geografici**, **data warehouse**…

Database - introduzione

- Un **database** è una raccolta di dati logicamente correlati
- Esempio: una rubrica telefonica
- Al giorno d'oggi, una rubrica telefonica può essere memorizzata da un database interno al telefono, in un PC tramite un foglio di calcolo, oppure tramite Microsoft Access, etc…
- Un database, per essere considerato tale, deve avere queste proprietà:
    1. non deve essere una raccolta di dati casuali privi di significato
    2. deve rappresentare un aspetto della **realtà**: ad esempio, i dati relativi alle analisi del sangue effettuate in un determinato ospedale
    3. deve essere progettato per un utilizzo da parte di qualcuno (persona) o qualcosa (applicazione software)
    4. i cambiamenti che avvengono nella realtà devono riflettersi nel database: ad esempio, l'arrivo di un nuovo dipendente in un'azienda deve riflettersi nell'inserimento dei dati relativi al dipendente all'interno del database aziendale

DBMS

- Un **DBMS** (***D**ata**b**ase **M**anagement **S**ystem*) è un sistema software estremamente complesso e *general-purpose*, che permette la creazione e la manutenzione di uno o più database. È *general purpose* perché permette di creare qualsiasi tipo di database
- Un DBMS permette:
  - la **definizione** del database, che consiste nello stabilire quali sono le strutture impiegate per memorizzare i dati e i vincoli che intercorrono tra di essi
  - la **costruzione** del database, ossia l'inserimento dei dati
  - la **manipolazione** del database, ossia la modifica dei dati precedentemente memorizzati
  - la **condivisione** del database tra diversi utenti e programmi

Modellazione di una realtà (*miniworld*)

- Un database per la gestione delle informazioni relative alla realtà scuola necessita sicuramente, tra le altre cose, di:
    - informazioni anagrafiche degli studenti
    - informazioni anagrafiche dei docenti e del personale ATA
    - informazioni relative ai dipartimenti e agli indirizzi
    - l'orario
    - la composizione dei consigli di classe
    - etc…

Modellazione di una realtà

- Possiamo già iniziare a descrivere le «cose» della realtà «scuola». Queste cose sono chiamate **entità** (*entities*): in questo caso, potremmo avere a che fare con le entità **studente**, **docente**, **classe**, **libro di testo**, **compito in classe**, etc…

- E se la realtà fosse un ospedale? Allora, le entità sarebbero: **paziente**, **personale medico**, **reparto**, **visita medica**, **referto**, etc…

- È evidente che le entità non saranno oggetti isolati, ma collegati

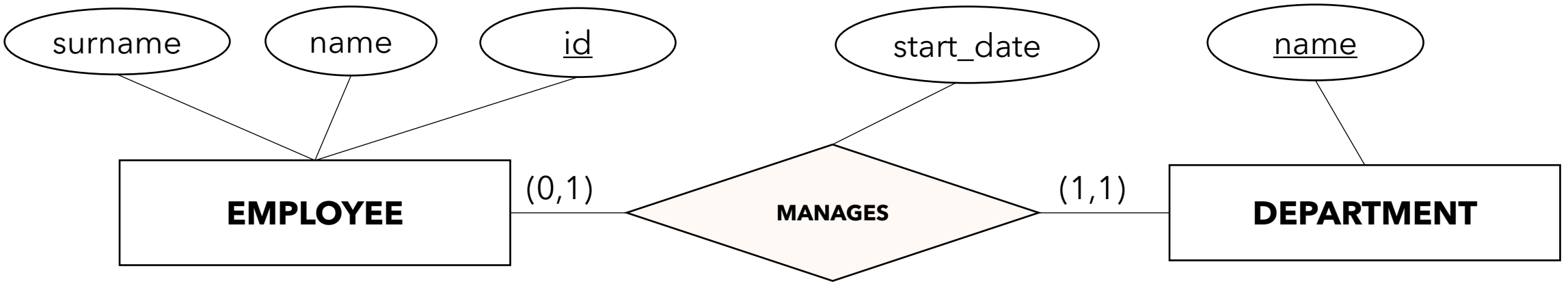
- Si dice che tra le entità sussistono delle **relazioni** (*relationships*)

Operazioni su un database

- Quali operazioni potremmo effettuare su un database scolastico?
  - cercare tutti gli studenti di cognome «Verdi»
  - cercare tutti gli studenti di una determinata classe
  - inserire un nuovo studente in una classe
  - cercare le anagrafiche di tutti i docenti di una determinata classe
  - cancellare gli insegnanti appena andati in pensione
  - etc…

Diagrammi ER – associazione di cardinalità 1:1

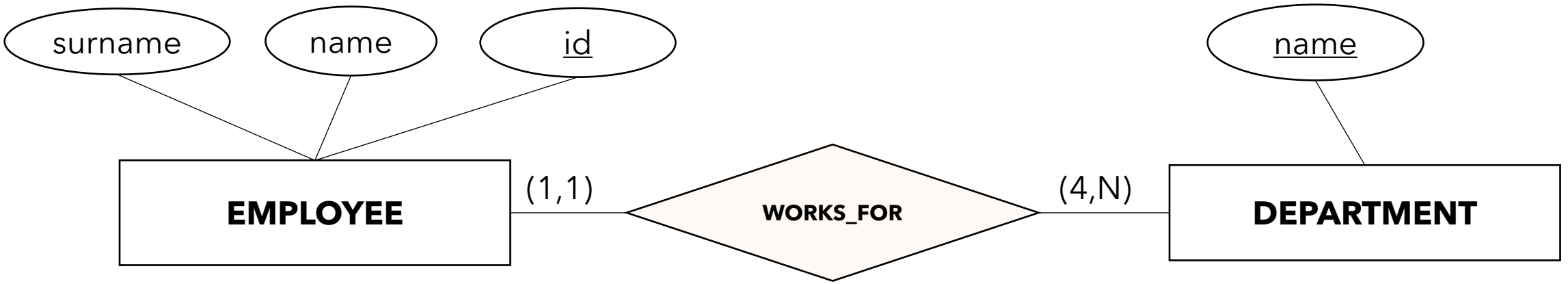
- Entità: EMPLOYEE, DEPARTMENT, associazione **one-to-one** (**1:1**)
- Un dipendente dirige min 0, max 1 reparto, a partire dalla data 'start_date'. Un reparto è diretto da min 1, max 1 dipendente.



NB: gli attributi sottolineati sono detti attributi chiave: ad esempio, non possono esistere 2 dipendenti con lo stesso id

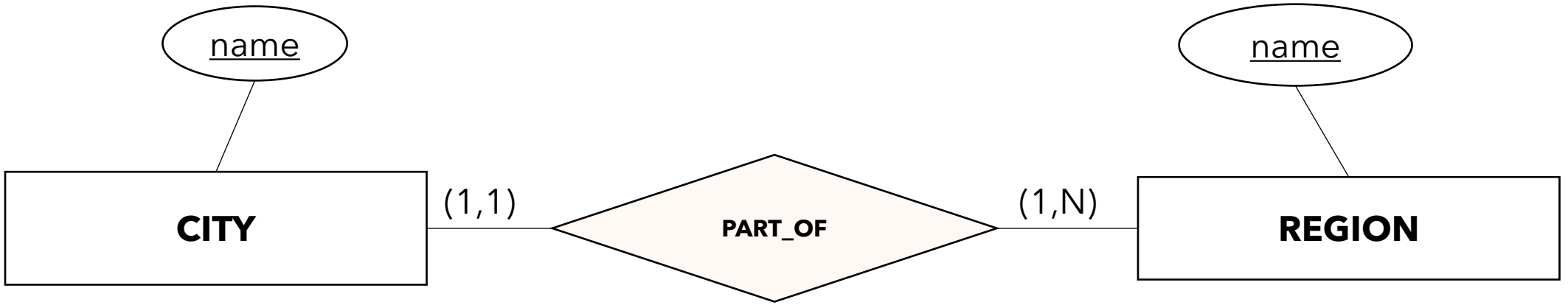
Diagrammi ER – associazione di cardinalità 1:N

- Entità: EMPLOYEE, DEPARTMENT, associazione **one-to-many** (**1:N**)
- Un dipendente lavora in min 1, max 1 reparto. In un reparto lavorano min 4, max N dipendenti.



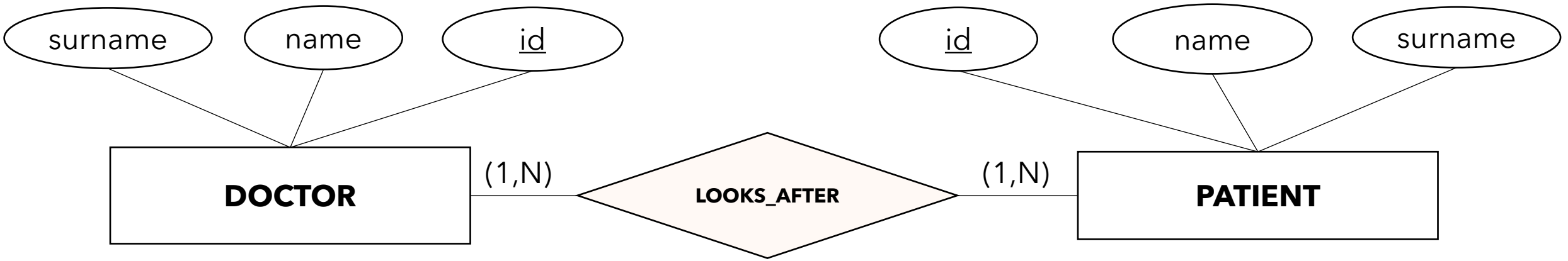
Diagrammi ER – associazione di cardinalità 1:N

- Entità: `CITY`, `REGION`, associazione **one-to-many** (**1:N**)
- `Una città appartiene a min 1, max 1 regione. Una regione comprende min 1, max N città.`



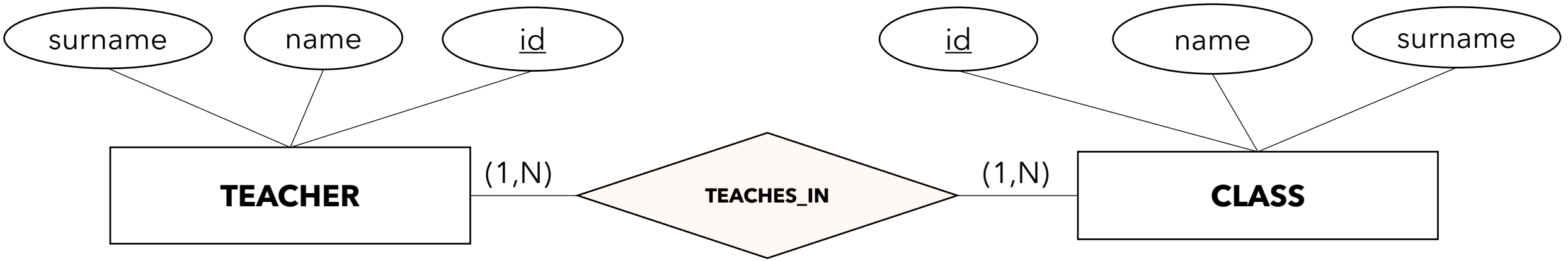
Diagrammi ER – associazione di cardinalità M:N

- Entità: `DOCTOR`, `PATIENT`, associazione **many-to-many** (**M:N**)
- `Un medico segue min 1, max N pazienti. Un paziente è seguito da min 1, max N medici.`



Diagrammi ER – associazione di cardinalità M:N

- Entità: TEACHER, CLASS, associazione **many-to-many** (**M:N**)
- Un insegnante insegna in min 1, max N classi. In una classe insegnano min 1, max N insegnanti.



Il modello relazionale

- Il **modello relazionale** rappresenta un database come insieme di relazioni
- Una relazione assomiglia ad una tabella
- Alcuni termini formali:
  - **relazione**: tabella
  - **tupla**: riga di una tabella
  - **attributo**: intestazione di colonna
  - **dominio**: il tipo dei valori possibili per una colonna
  - **stato di una relazione**: insieme delle tuple di una relazione

Il modello relazionale

- **Esempio**: schema di relazione di grado 7 (o arità 7) (i.e. con 7 attributi) per memorizzare le informazioni relative agli studenti di un'università:
  - `STUDENT(ssn, name, surname, address, birthdate, GPA)`
- La relazione precedente può essere espressa specificando i tipi degli attributi, in questo modo:
  - `STUDENT(ssn:string, name:string, surname:string, address:string, birthdate:date, GPA:real_number)`
- **NB**: gli attributi sottolineati sono gli attributi chiave

Il modello relazionale

STUDENT
(relation name)

	ssn	name	surname	address	birthdate	GPA
tuple	111-12-1111	john	black	3 Italy Lane	05/03/2001	3.25
tuple	211-12-3333	donald	white	4 California Lane	12/23/2001	4.25
tuple	131-12-2111	mary	gray	7 London Lane	11/15/2002	NULL

NB: il valore `NULL` ha il significato di *valore ignoto*, o di *valore non applicabile alla tupla*, oppure *valore esistente ma non disponibile*…

Il modello relazionale

- Le relazioni del modello relazionale possono rappresentare:
  - entità
  - associazioni
- Generalmente, in un database relazionale, esistono diversi **vincoli** (*constraints*) sui valori presenti nelle relazioni:
  - vincoli di dominio (*domain constraints*)*:* tipi degli attributi
  - vincoli di chiave (*key constraints*)
  - vincoli sul valore `NULL`: specificano se per un attributo il valore `NULL` è ammesso o meno

Il modello relazionale

- **Superkey** (*superchiave*): sottoinsieme di attributi SK di una relazione R per il quale, per ogni coppia di tuple distinte t_1 e t_2, $t_1[SK] \neq t_2[SK]$
- **Key** (*chiave*): una chiave **k** di uno schema di relazione R è una superchiave per la quale se si toglie un attributo da k, trasformando k in k', allora k' non è più una superchiave di R (k *è una superchiave minima*)
- **Candidate key** (*chiave candidata*): possibile chiave di uno schema di relazione
- **Primary key** (*chiave primaria*): chiave candidata scelta, i cui valori permettono di identificare le tuple di una relazione

Il modello relazionale

- Uno **schema di database relazionale** è formato da un insieme di relazioni e da un insieme di **vincoli di integrità** (*integrity constraints*)
- Esempi:
  - `HOSPITAL = {DOCTOR, PATIENT, EXAM, WARD}`
  - `COMPANY = {EMPLOYEE, DEPARTMENT, …}`

Il modello relazionale

- Considerare le relazioni seguenti del database COMPANY:
  - EMPLOYEE(ssn, fname, lname, birthdate, dept_num)
  - DEPARTMENT(num, name)
- Gli attributi dept_num di EMPLOYEE e num di DEPARTMENT hanno entrambi il significato di *codice identificativo di un reparto*
- In particolare, dept_num indica il reparto in cui lavora un dipendente rappresentato in una tupla della relazione EMPLOYEE

Il modello relazionale

EMPLOYEE

ssn	fname	lname	birthdate	dept_num
111-12-1111	john	black	05/03/2001	1
211-12-3333	donald	white	12/23/2001	3
131-12-2111	mary	gray	11/15/2002	3

Il modello relazionale

- **Entity integrity constraint**: il valore di una primary key non può essere `NULL`
- **Referential integrity constraint** (*vincolo di integrità referenziale*): sussiste tra 2 relazioni; permette di mantenere coerenti le tuple delle 2 relazioni. Informalmente: una tupla di una relazione R_1 che fa riferimento ad una relazione R_2, deve far riferimento ad una tupla esistente di R_2. Esempio: il valore dell'attributo `dept_num` di `EMPLOYEE` deve riferirsi al valore di `num` di una tupla di `DEPARTMENT`

Il modello relazionale

- **Foreign key** (*chiave esterna*): le condizioni di una chiave esterna specificano un vincolo di integrità referenziale tra 2 schemi di relazione R_1 e R_2:
  - un insieme di attributi FK di una relazione R_1 è una foreign key di R_1 che si riferisce (*references*) ad una relazione R_2 se:
    - gli attributi di FK hanno gli stessi domini degli attributi della primary key (PK) di R_2
    - se t_1 è una tupla dello stato di R_1, allora $t_1[FK] = t_2[PK]$
- I vincoli di integrità referenziale (e le chiavi esterne) derivano generalmente dalle associazioni tra le entità dello schema concettuale

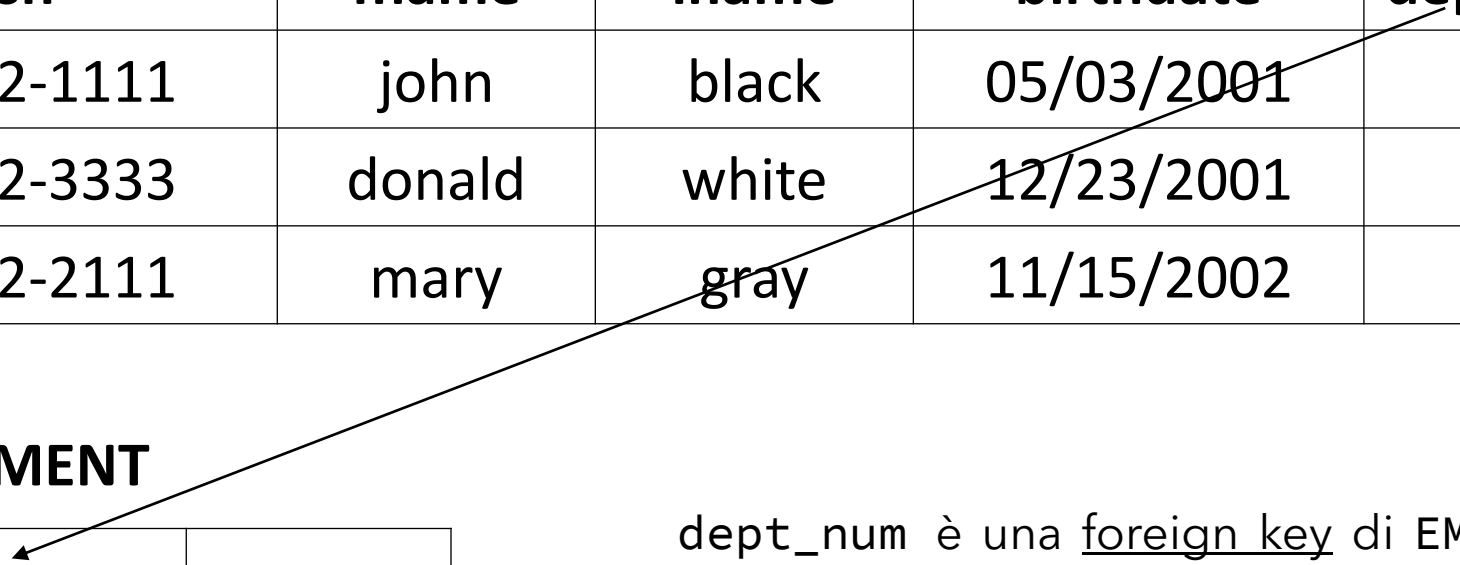
Il modello relazionale

EMPLOYEE

ssn	fname	lname	birthdate	dept_num
111-12-1111	john	black	05/03/2001	1
211-12-3333	donald	white	12/23/2001	3
131-12-2111	mary	gray	11/15/2002	3

DEPARTMENT

num	name
1	HR
3	IT



`dept_num` è una foreign key di `EMPLOYEE` che si riferisce alla relazione `DEPARTMENT` (`references DEPARTMENT`)

se l'attributo `dept_num` di `EMPLOYEE` non ha valore `NULL`, allora deve corrispondere al valore della `PK num` di una tupla di `DEPARTMENT`

Il modello relazionale

EMPLOYEE

ssn	fname	lname	birthdate	dept_num
111-12-1111	john	black	05/03/2001	1
211-12-3333	donald	white	12/23/2001	3
131-12-2111	mary	gray	11/15/2002	3

DEPARTMENT

num	name
1	HR
3	IT

NB: la cancellazione (`DELETE`) di una tupla di `DEPARTMENT` può causare una violazione dell'integrità referenziale.
In caso di violazione, le opzioni sono:

- restrict: la cancellazione della tupla non viene eseguita
- cascade: la tupla viene cancellata; il DBMS prova a cancellare anche le tuple che si riferiscono alla tupla cancellata
- set default: i valori degli attributi che si riferiscono alla tupla cancellata vengono posti a `NULL`, o a un valore di default