DAL MODELLO CONCETTUALE AL MODELLO LOGICO:
ALGORITMI DI DERIVAZIONE

Roadmap

1. Database Fundamentals
2. Database Design
3. Relational Model
4. Derivation Rules
5. SQL Language Fundamentals

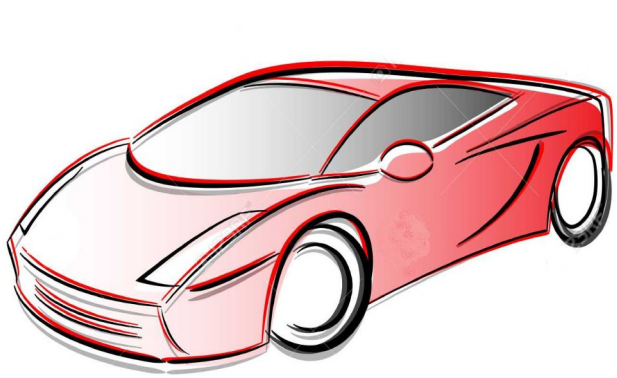
Focus: nomenclatura

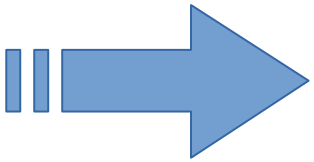
- regole di derivazione
- regole di conversione
- algoritmi di conversione
- algoritmi di derivazione
- derivation rules

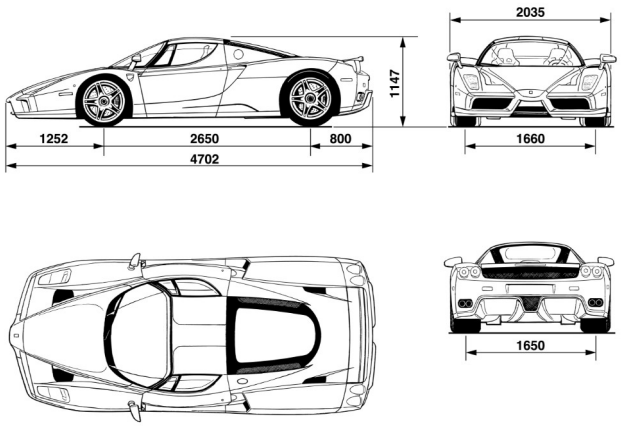
SONO TUTTI SINONIMI

Regole di derivazione

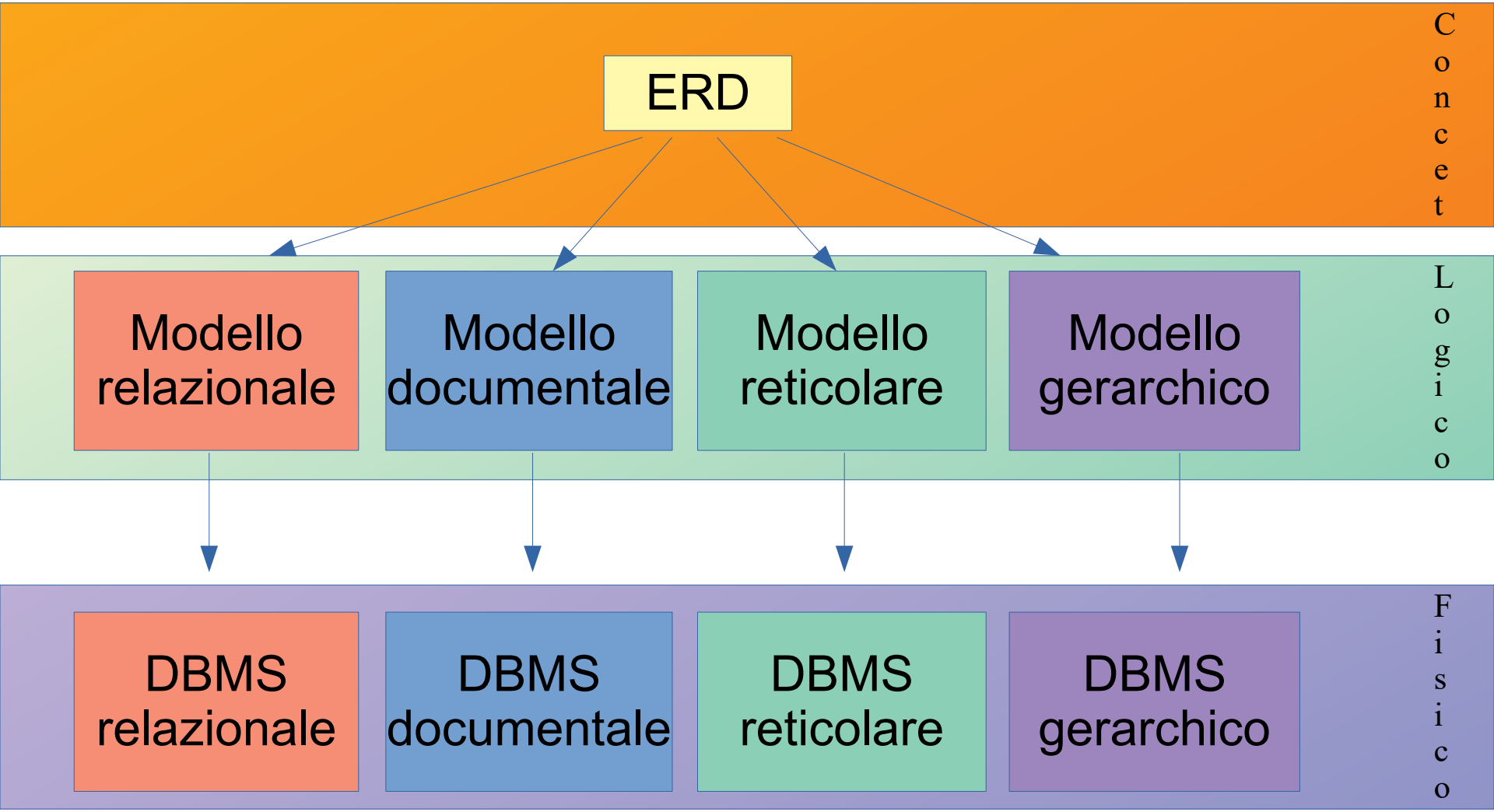
Le regole di derivazione (algoritmi di conversione) ci permettono di passare dal modello concettuale (mediante il quale è stata progettata la base di dati) al modello logico



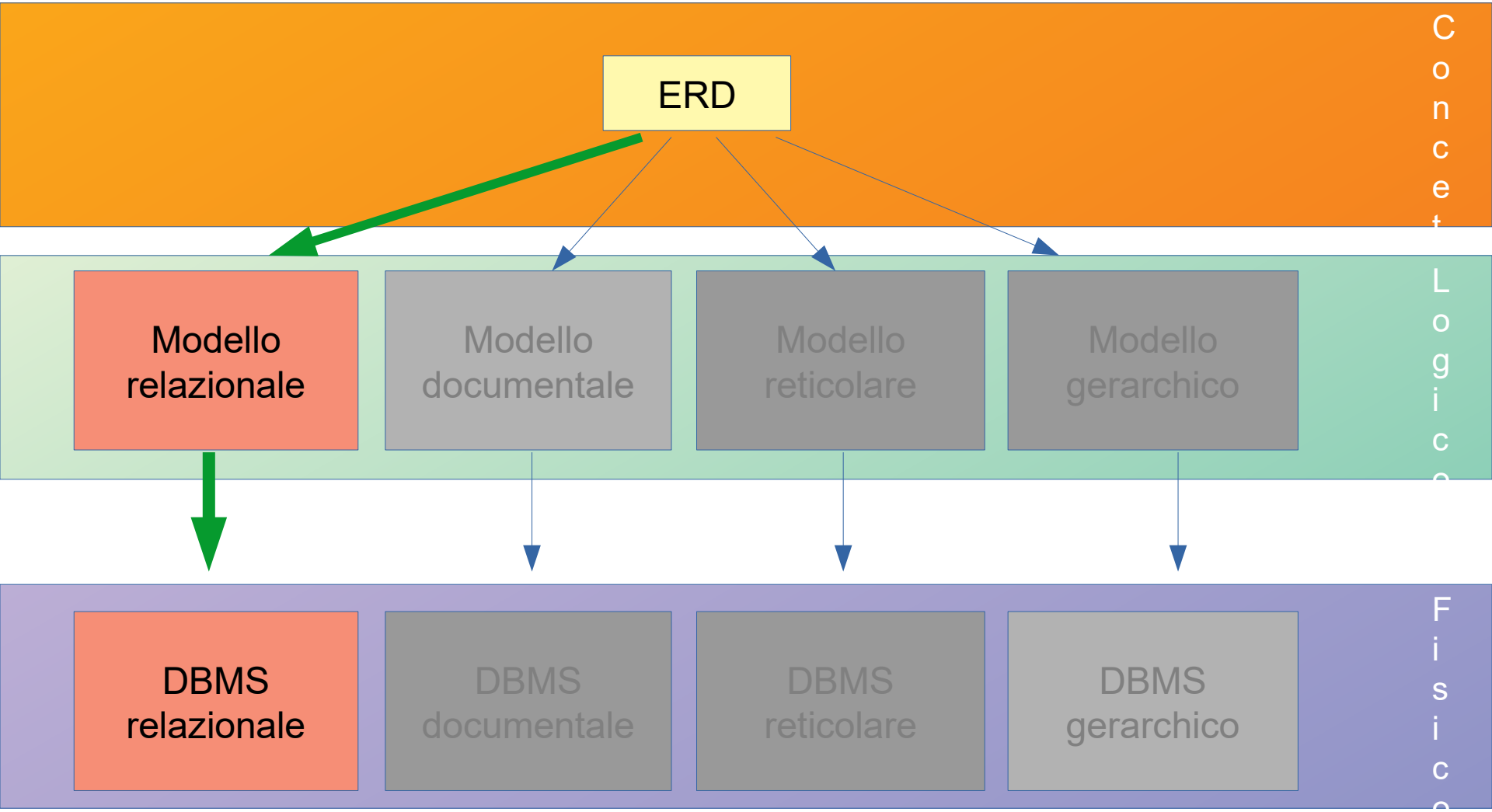




Modelli di progettazione



Modelli di progettazione



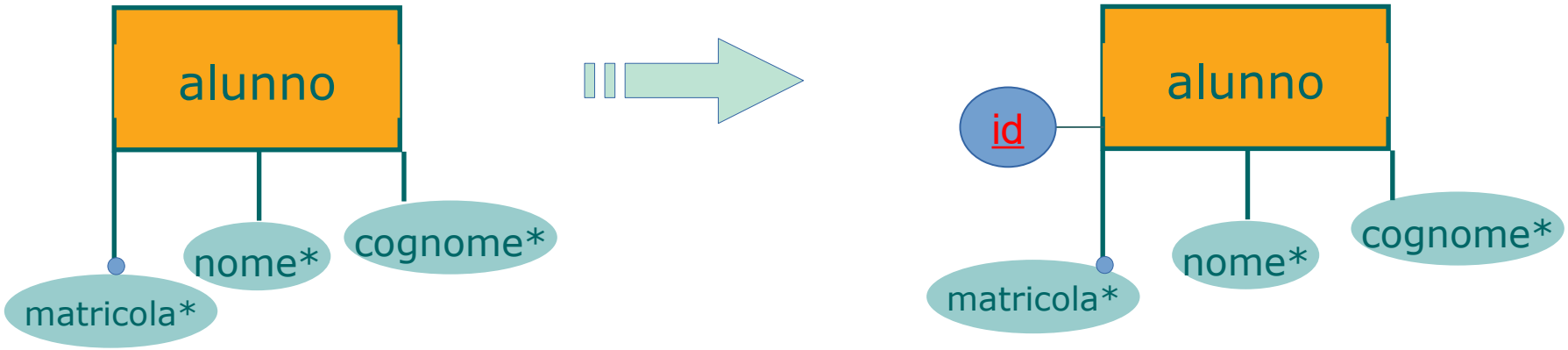
Modello ER → Modello relazionale

- Il Modello Relazionale può essere ricavato direttamente dal Modello ER, attraverso una sequenza di operazioni di conversione (o derivazione).

- Le «Regole di derivazione» servono per passare dalla fase di progettazione del «livello concettuale» a quella del «livello logico», cioè per trasformare lo schema E-R nello schema delle relazioni (insieme di tabelle).

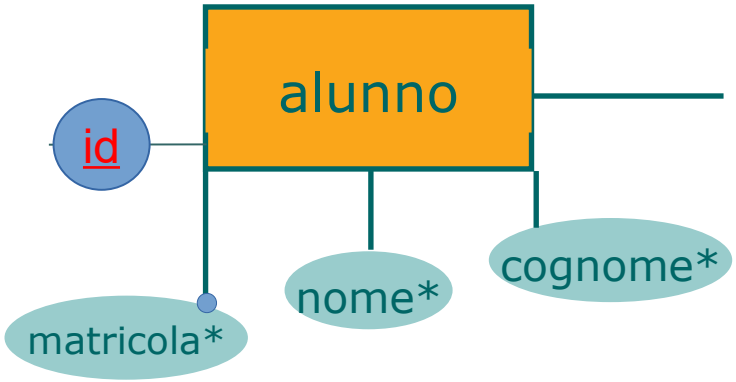
Regola 1: Primary Key

- Prima di procedere con la CONVERSIONE, dal punto di vista informatico, è conveniente aggiungere ad ogni entità un attributo di tipo **id** (numerico intero positivo e non nullo) come identificatore. Tale attributo diventerà **la** chiave primaria **della tabella che ne deriveremo.**
- Notazioni per la Primary Key:
  - Sottolineatura
  - "PK" come apice



Regola 2: entità→ tabelle

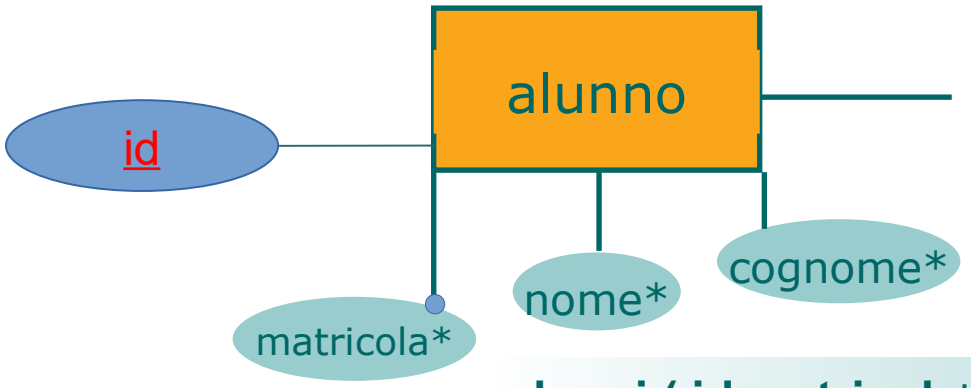
Ogni entità dello schema E-R viene trasformata in una relazione(tabella)
Gli attributi della tabella saranno gli attributi dell'entità
Notazione: nome_tabella(lista_attributi)



alunni (id,matricola^{u*}, nome*, cognome*

Regola 3: vincoli

1) Ogni attributo di un'entità (incluso il campo id) diventa il nome di una delle colonne della tabella che ne è stata derivata.
2) L'attributo id è la chiave primaria
3) Sugli identificatori univoci va posto un vincolo di unicità (contrassegnandoli con una **u**)
4) Sotto agli identificatori univoci multipli va posto un archetto
5) Sugli attributi obbligatori va posto un vincolo di esistenza (contrassegnandoli con un *****)



alunni (id,matricola^{u*}, nome*, cognome*

Focus: le prime tre regole

1) Aggiungo id a tutte le entità
2) Ogni entità diventa una tabella (copio attributi)
3) Impongo i vincoli di unicità, esistenza e primary key agli attributi)

Relazioni **uno a molti**

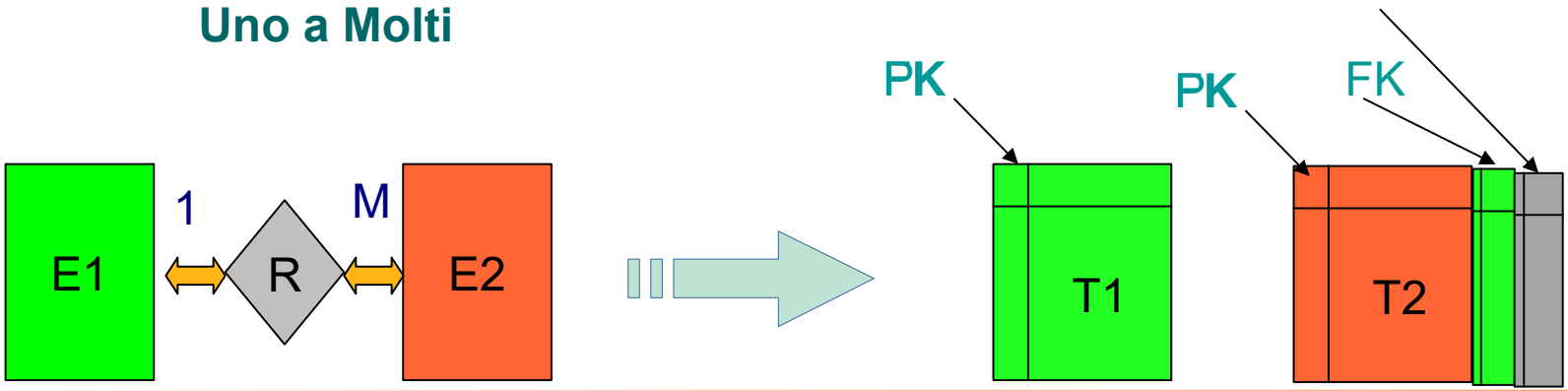
Per ogni relazione di tipo 1:M si aggiunge una colonna alla tabella dal lato M i cui valori esistono come **chiave primaria** nella tabella del lato **1**

Questa nuova colonna sarà una chiave esterna (foreign key).

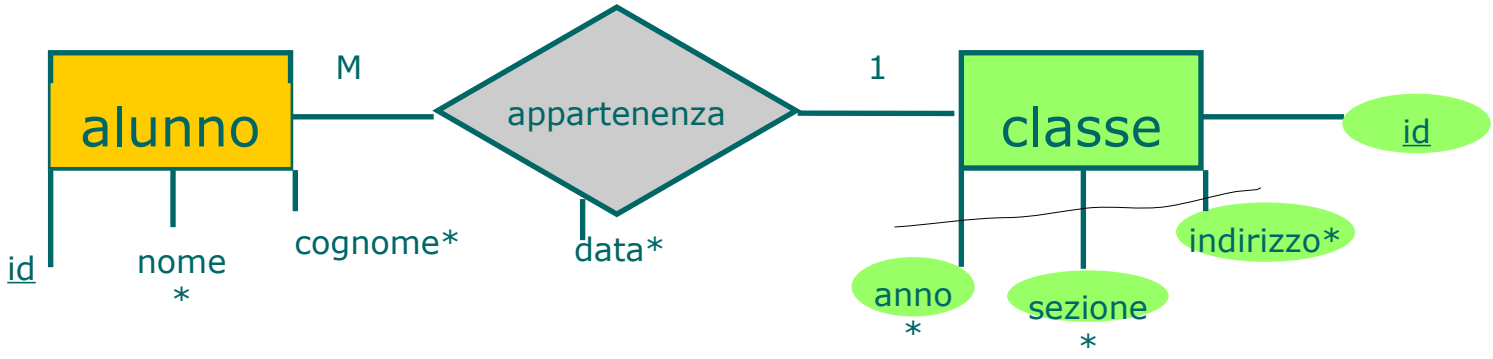
Notazione per la Foreign Key

- Nome FK: id_<nome tabella lato 1>_<*nome relazione*>
- sopralineata
- FK come apice

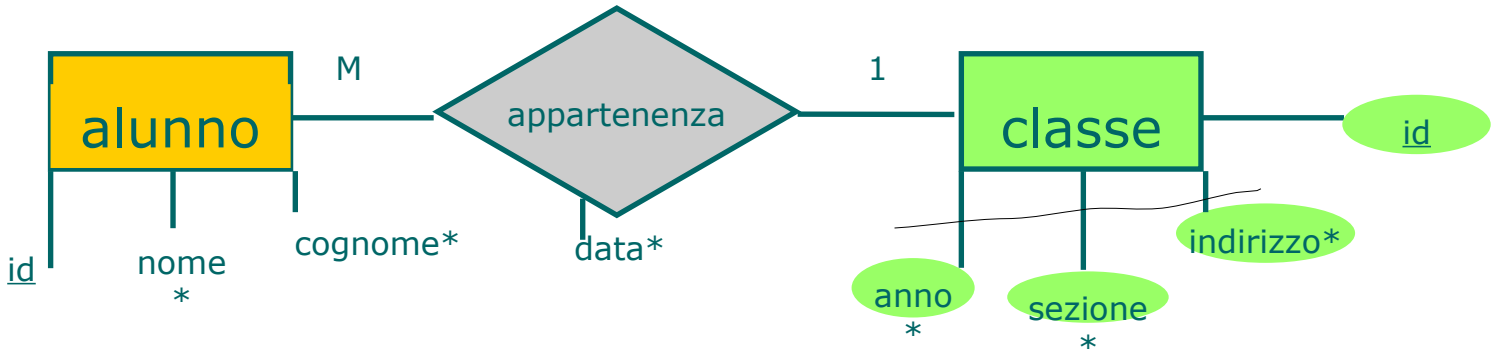
Eventuali attributi di relazione (grigio) saranno colonne della tabella dal lato M



Relazioni uno a molti (esempio)



Relazioni uno a molti (esempio)



E' necessario aggiungere alla tabella **«alunni»** un **nuovo attributo,** a cui si darà il nome **id_classe** che sarà collegato all'attributo **id** della tabella «classi».

classe (id, anno*, sezione*, indirizzo*)

u

alunno (id, nome*, cognome*, id_classe ,data*)

Istanza logica

città

id	nome	kmq
3	Roma	120
5	Milano	80
6	Ladispoli	20

abitante

id	nome	cognome	id_città
150	Rino	Rano	6
151	Nino	Rossi	3
152	Gino	Verdi	5

Relazioni?
Attributi?
Tuple?

Spoiler: schema fisico

ABITANTI –<RESIDENZA>-CITTA'

```
create table città(
id int unsigned primary key auto_increment,
nome varchar(20) unique not null,
kmq int);

create table abitanti(
id int unsigned primary key auto_increment,
nome varchar(50),
id_città int unsigned,
indirizzo varchar(60) not null,
foreign key(id_città) references città(id)
);
```

Esercizio: classi

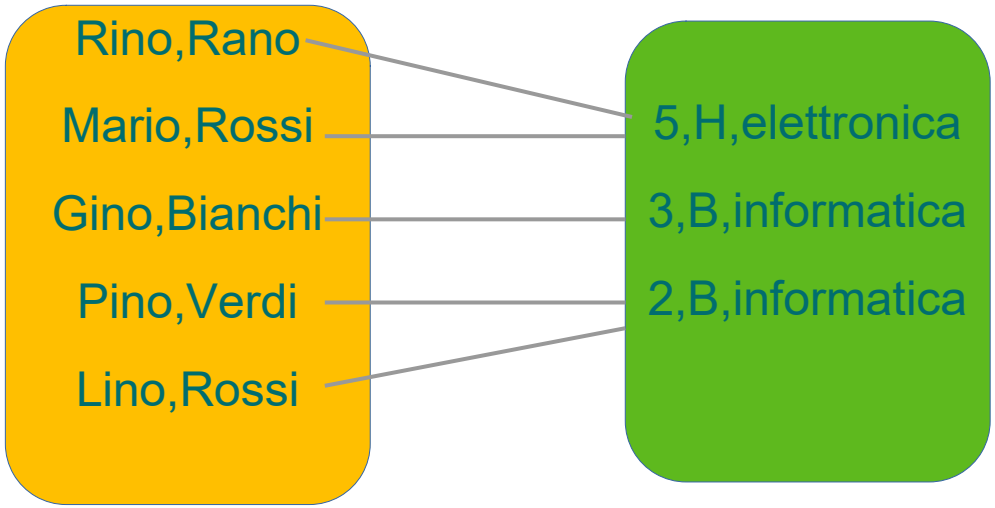
Dato lo schema logico

`classe ( id, anno*, sezione*, indirizzo* )`

u

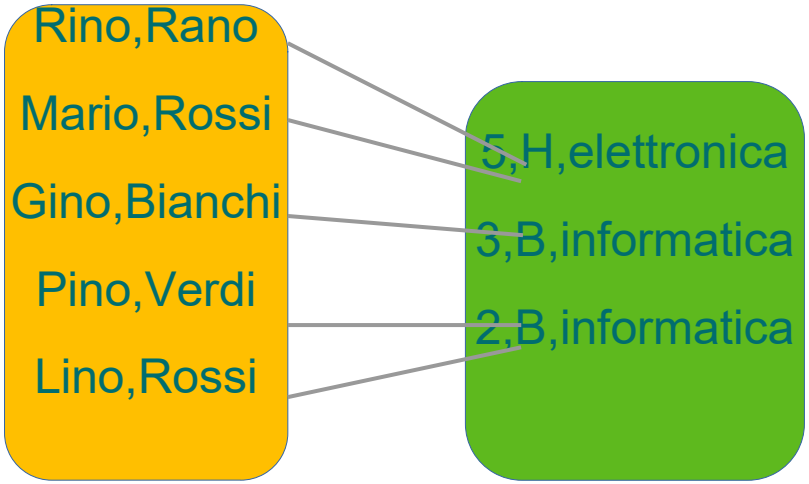
`alunno ( id, nome*, cognome*, id_classe )`

1) Disegnare le tabelle
2) Popolare le tabelle iscrivendo 5 alunni a 3 diverse classi



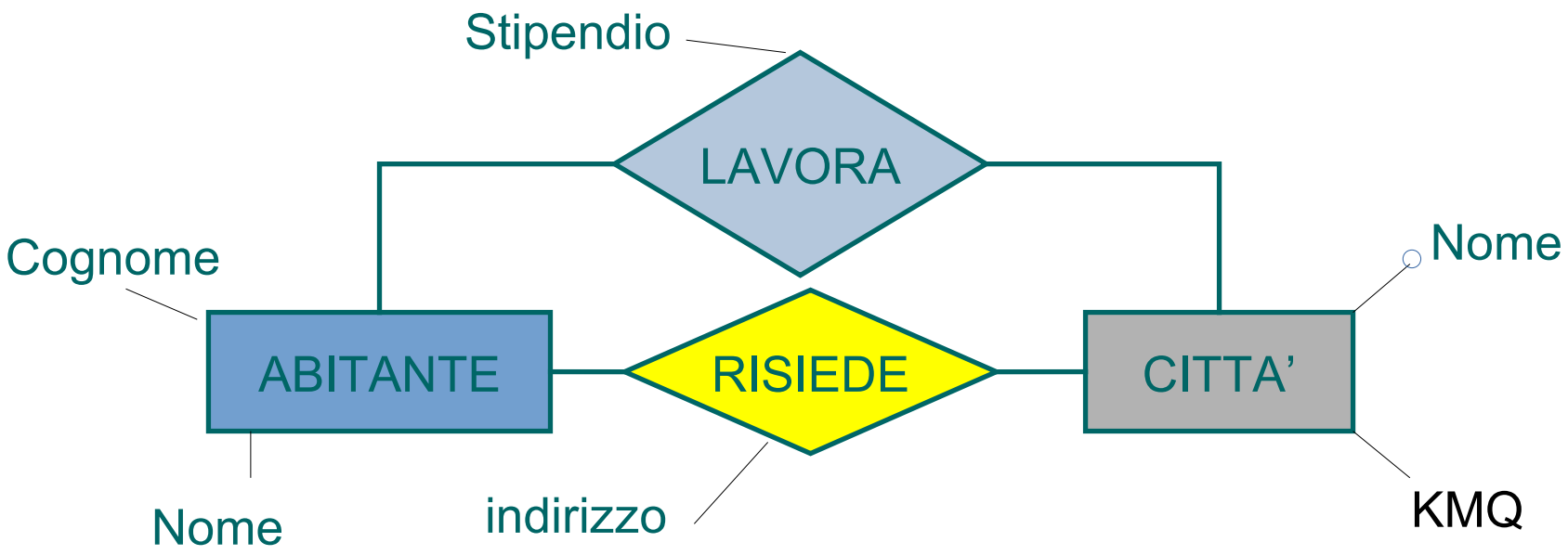
Esercizio: classi

ID	ANNO	SEZ	INDIRIZZO
1	5	H	elettronica
2	3	B	informatica
88	2	B	informatica



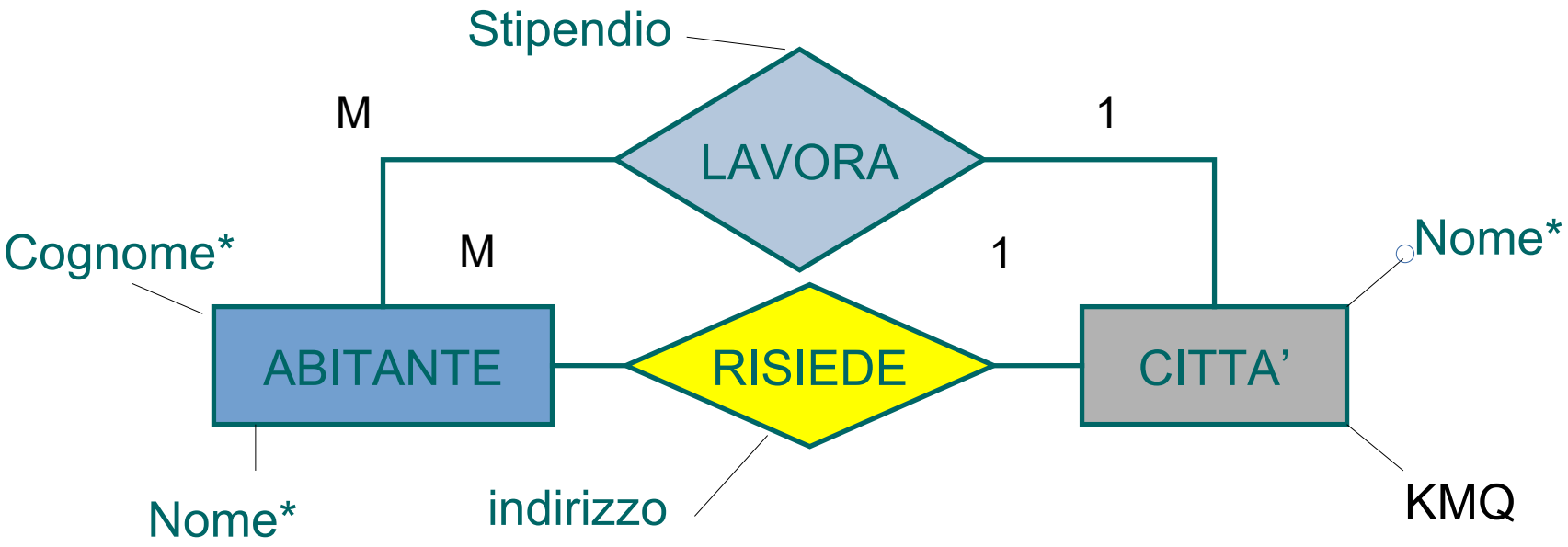
ID	NOME	COGNOME	ID_CLASSE
1	Rino	Rano	1
2	Mario	Rossi	1
3	Gino	Bianchi	2
4	Pino	Verdi	88
53	Lino	Rossi	88

Esercizio: residenza e lavoro



1) Completare il diagramma ER (vincoli, tipi di relazione, cardinalità)
   - -la residenza è obbligatoria
   - -gli abitanti hanno al più una sede di lavoro
2) Derivare lo schema logico
3) Ragionare sui vincoli di esistenza per le FK

Esercizio: residenza e lavoro



SOLUZIONE:

CITTA(id, nome*[u], kmq)
ABITANTI(id, nome*, cognome*,id_città_residenza*,id_città_lavoro,stipendio,indirizzo*)

Esercizio: residenza e lavoro

Dato lo schema

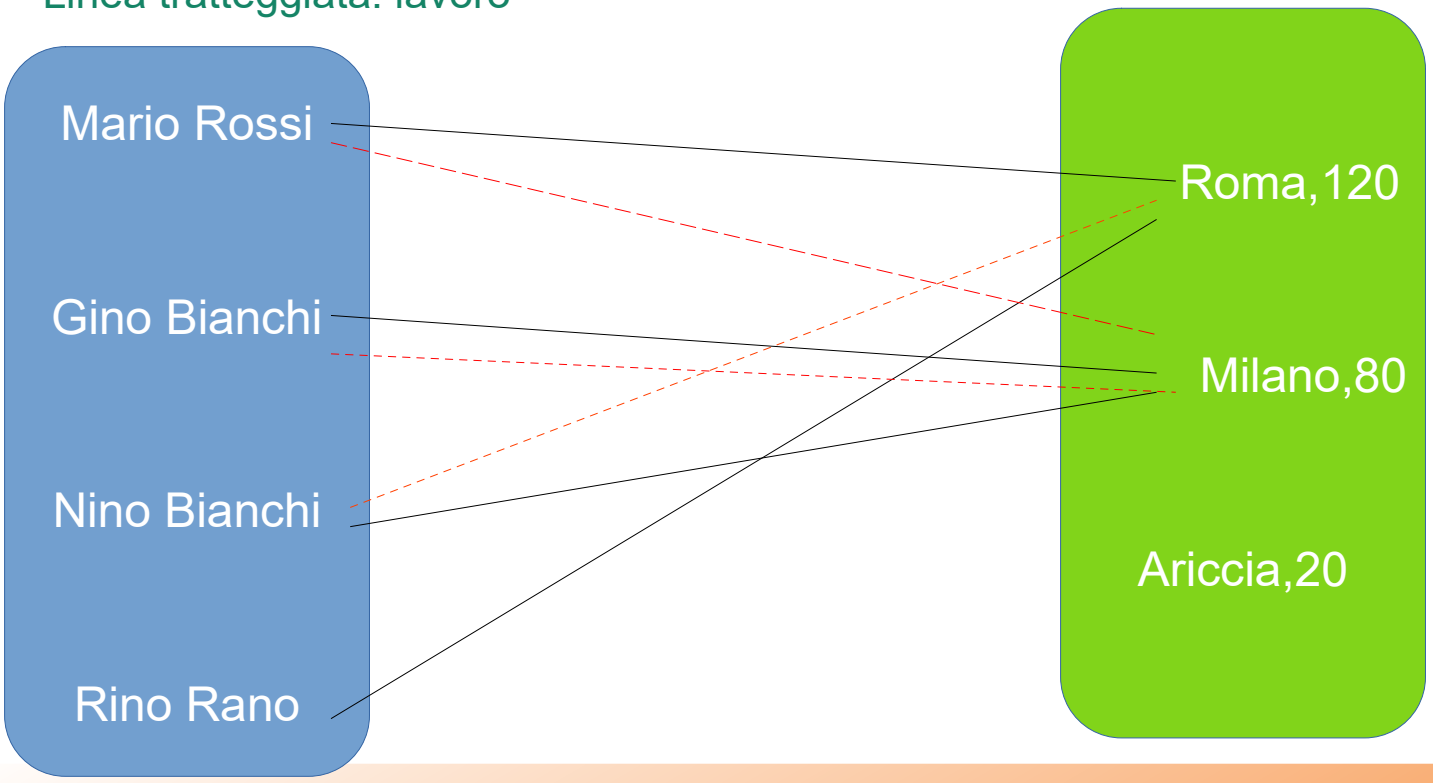
CITTA(id, nome*[u], kmq)
ABITANTI(id, nome*, cognome*,id_città_residenza*,id_città_lavoro,stipendio,indirizzo*)

Popolarlo con 3 città e 4 abitanti dove
-Un abitante non è pendolare
-Rino Rano è disoccupato

Esercizio: residenza e lavoro

SCHEMA LOGICO
CITTA(id, nome*[u], kmq)
ABITANTI(id, nome*, cognome*,id_città_residenza*,id_città_lavoro,stipendio,indirizzo*)

ISTANZA CONCETTUALE
Linea continua: residenza
Linea tratteggiata: lavoro



Esercizio: residenza e lavoro

SOLUZIONE:

CITTA(id, nome*, kmq)

ABITANTI(id, nome*, cognome*,id_città_residenza*,id_città_lavoro,stipendio,indirizzo*)

ID	NOME	KMQ
1	Roma	120
2	Milano	80
5	Ariccia	20

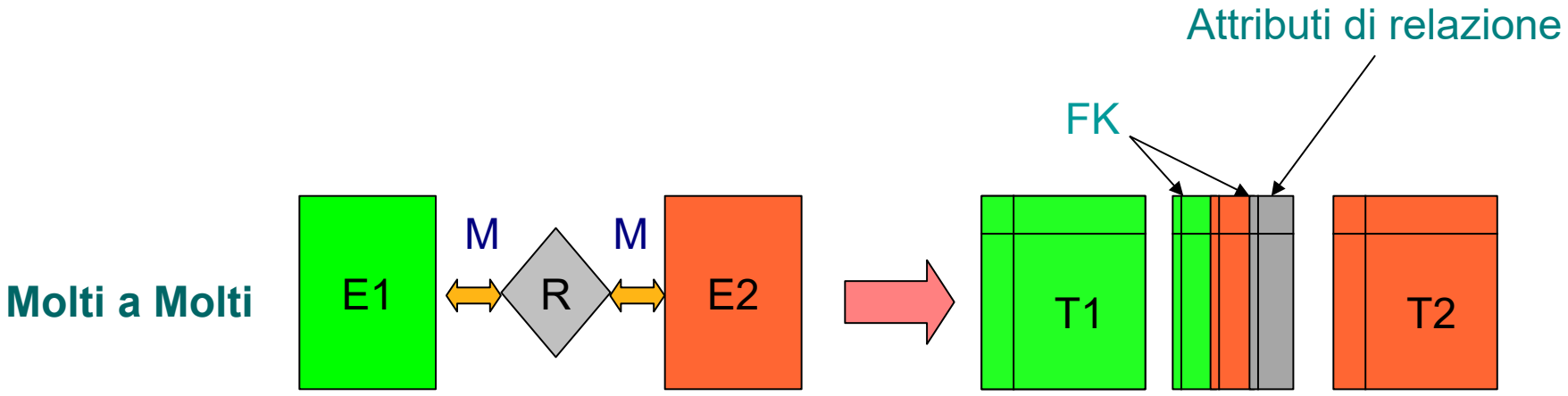
ID	NOME	COGNOME	ID_CITTA'_RES	ID_CITTA'_LAV	STIPENDIO	INDIRIZZO
1	Mario	Rossi	1	2	1300	Via Verdi
2	Gino	Bianchi	2	2	1500	Parco Sempione
3	Nino	Bianchi	2	1	1200	Navigli
4	Rino	Rano	1	null	null	Via Nazionale

Relazioni **molti a molti**

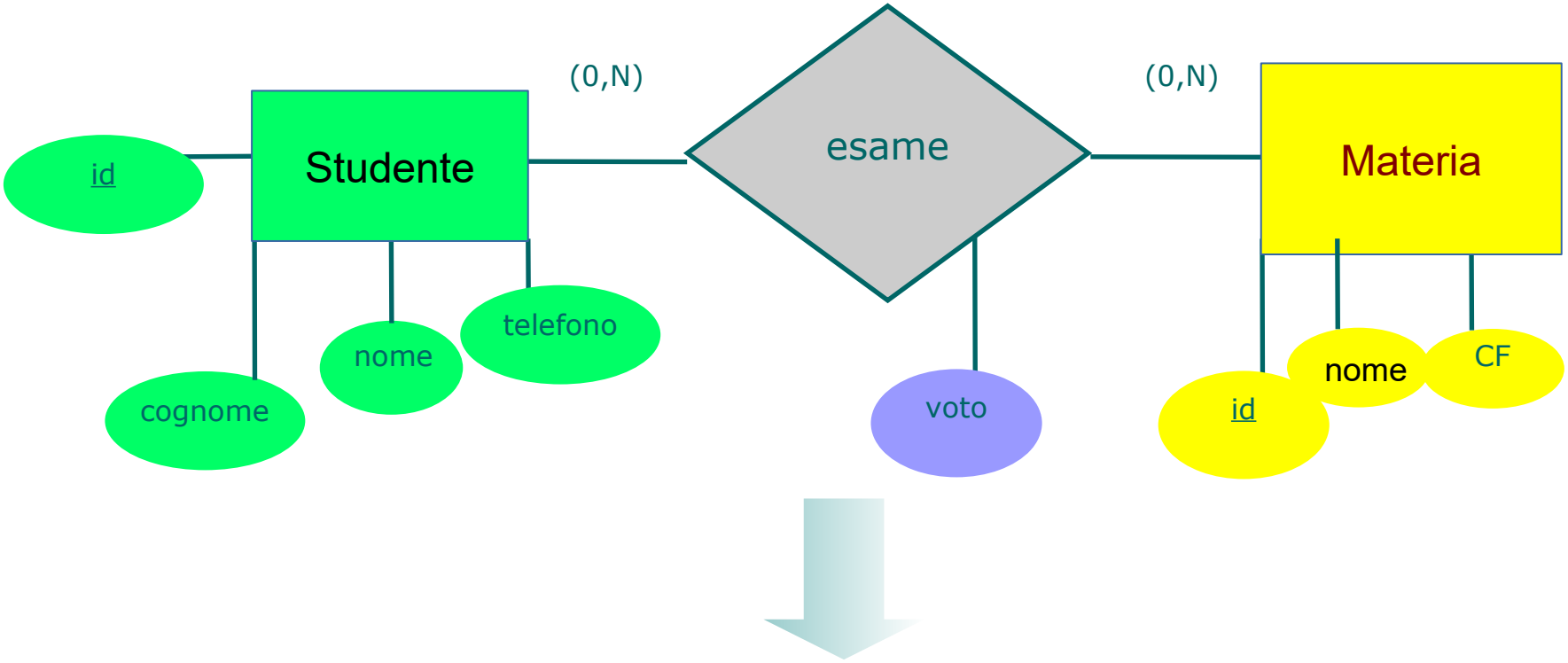
Per ogni relazione M a M tra due entità si aggiunge, alla due tabelle derivate dalle due entità, una **terza** tabella i cui attributi sono chiavi esterne collegate alle chiavi primarie delle prime due tabelle.

NOTE

- La nuova tabella è detta “tabella di collegamento”
- Eventuali attributi di relazione vanno nella tabella di collegamento
- Le FK avranno il vincolo di NOT NULL

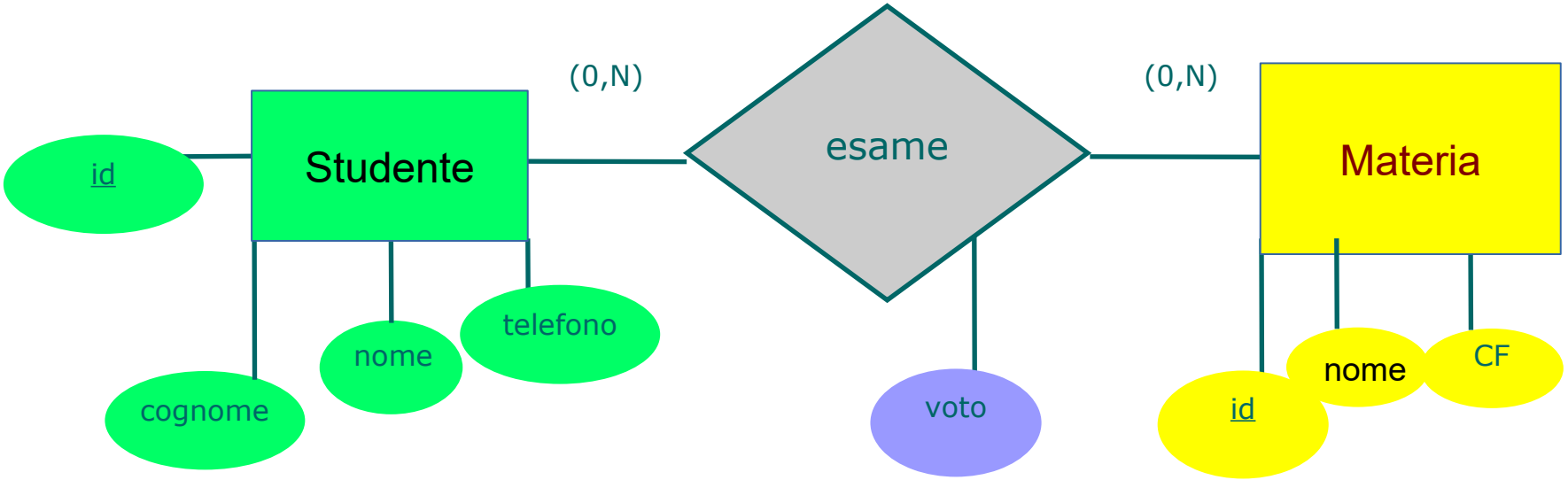


Relazioni Molti a Molti(esempio)



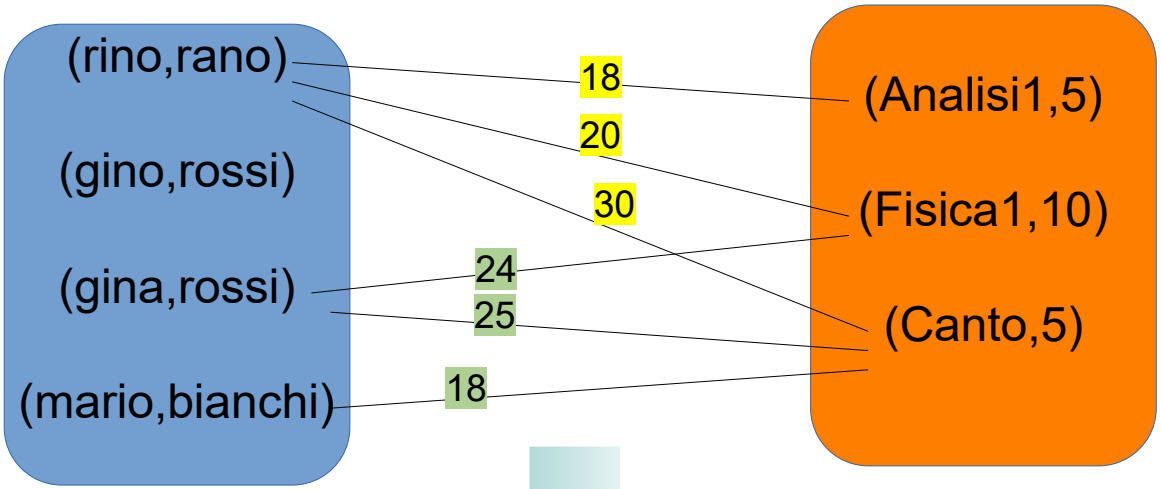
Derivare lo schema logico

Relazioni Molti a Molti (esempio)



Studenti(<u>id</u>,cognome,nome,telefono)
Materie(<u>id</u>,nome,CF)
Esami(<u>id</u>, id_studente, id_materia,voto)

Esercizio: Popolare lo schema logico



Studenti(id,cognome,nome,telefono)
Materie(id,nome, CF)
Esami(id,id_studente*, id_materia*,voto)

Popolare lo schema logico con 4 studenti, 3 materie, 6 esami

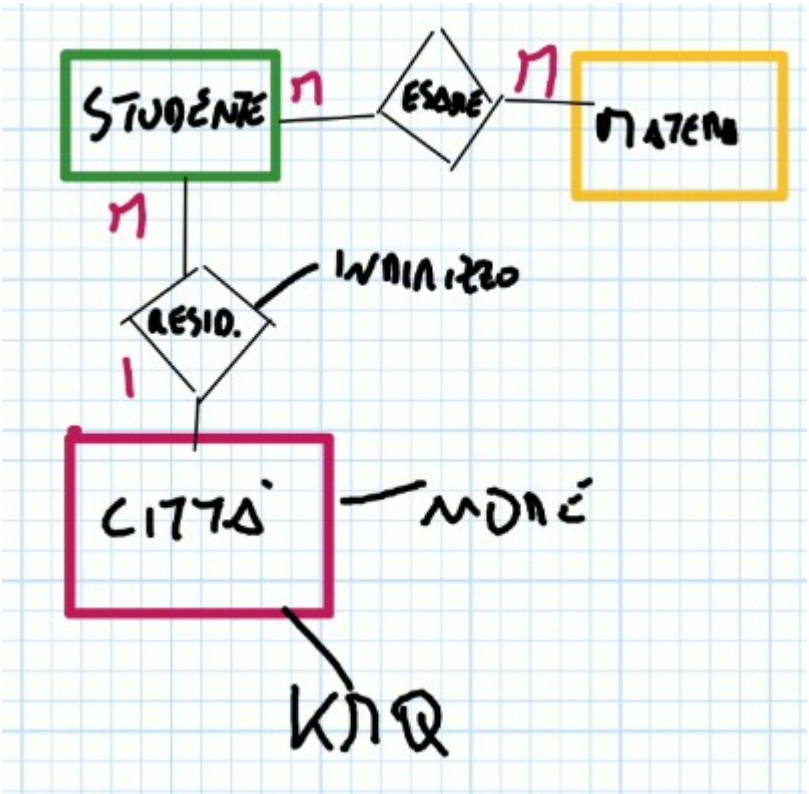
Soluzione: Popolare lo schema logico

id	nome	cognome	telefono
1	rino	rano	338
2	gino	rossi	333
3	gina	rossi	null
4	mario	bianchi	335

id	nome	CF
1	Analisi1	5
2	fisica1	10
3	canto	5

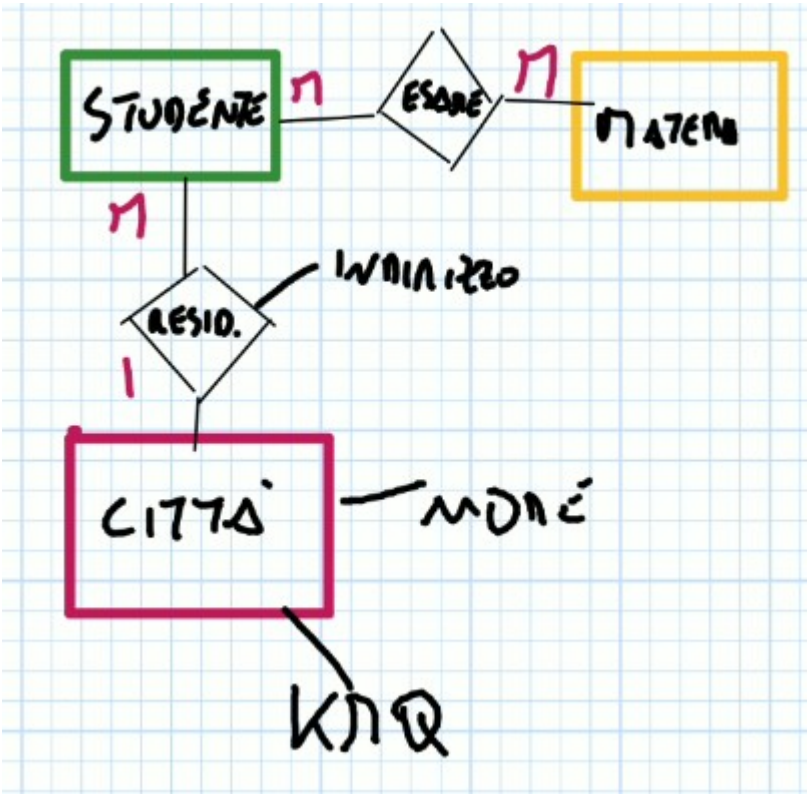
id	id_stud	id_mat	voto
1	1	1	18
2	1	2	20
3	3	2	24
4	3	3	25
5	1	3	30
6	4	3	18

Modificare il diagramma precedente aggiungendo l'entità CITTA'(nome*,kmq), la relazione RESIDENZA(indirizzo) e derivare il nuovo schema logico

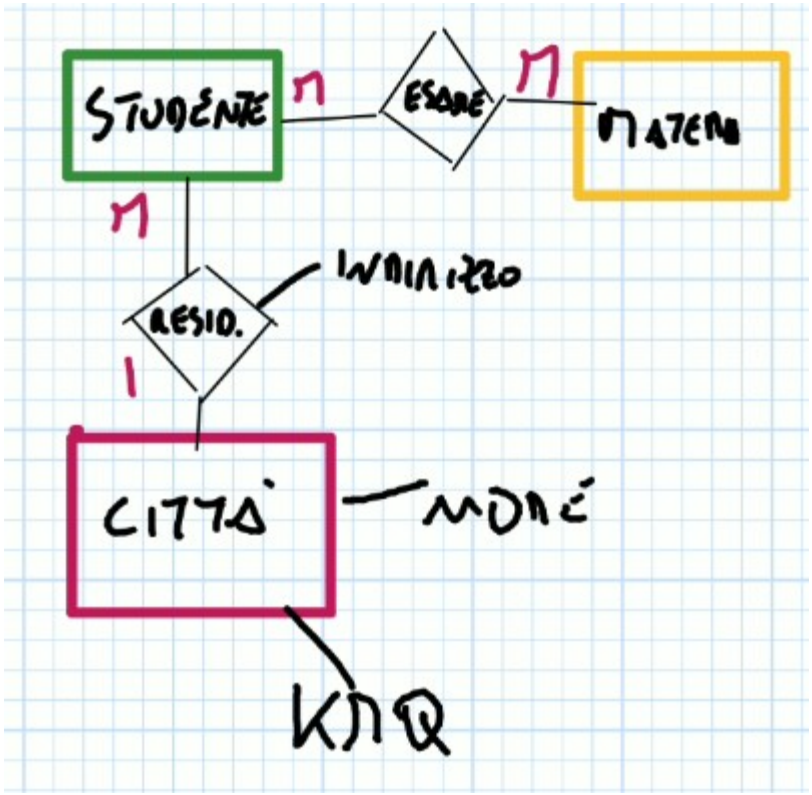


Esercizio: cambiamento dei requisiti

Studenti(id,cognome,nome,telefono)
Materie(id,nome, CF)
Esami(id,id_studente*, id_materia*,voto)



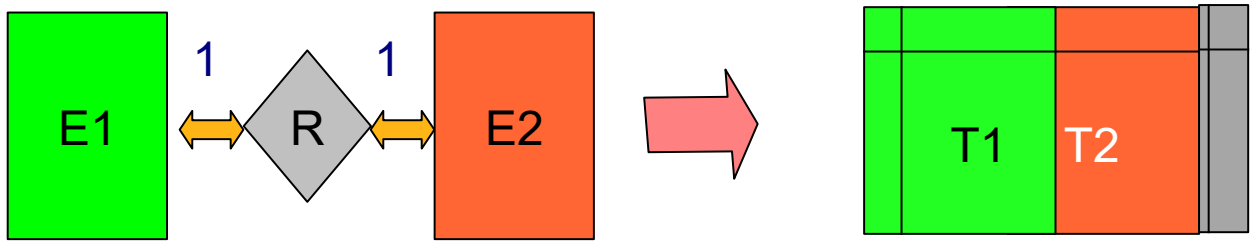
Studenti(id,cognome,nome,telefono,**id_città_res*,indirizzo_res***)
Materie(id,nome, CF)
Esami(id,id_studente*, id_materia*,voto)
Città(id,nome,kmq)



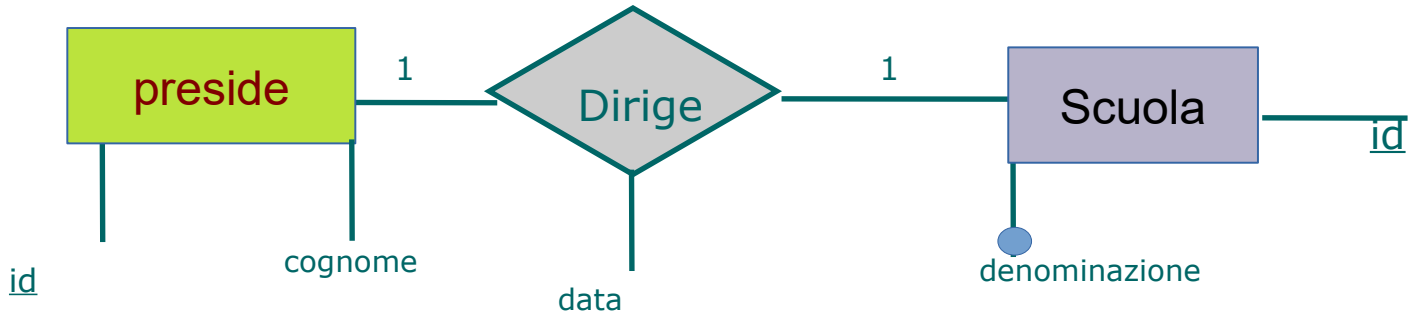
Relazioni **uno ad uno**

- Per ogni associazione binaria di tipo 1:1 si crea un'unica tabella che ha come attributi tutti gli attributi delle due entità
- **Note**
  - La tabella ottenuta ha un solo id (elimino l'altro)
  - Attributi di relazione vanno nell'unica tabella risultante
  - Per le relazioni 1-1 **non** ho necessità di foreign key

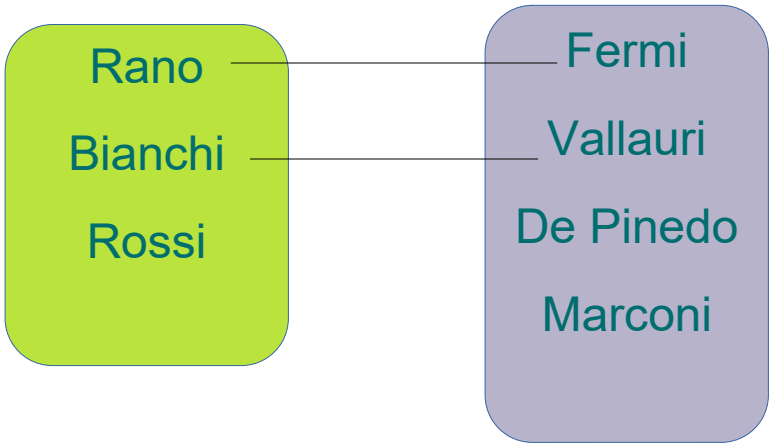
Uno a Uno



Relazioni uno ad uno (esempio)

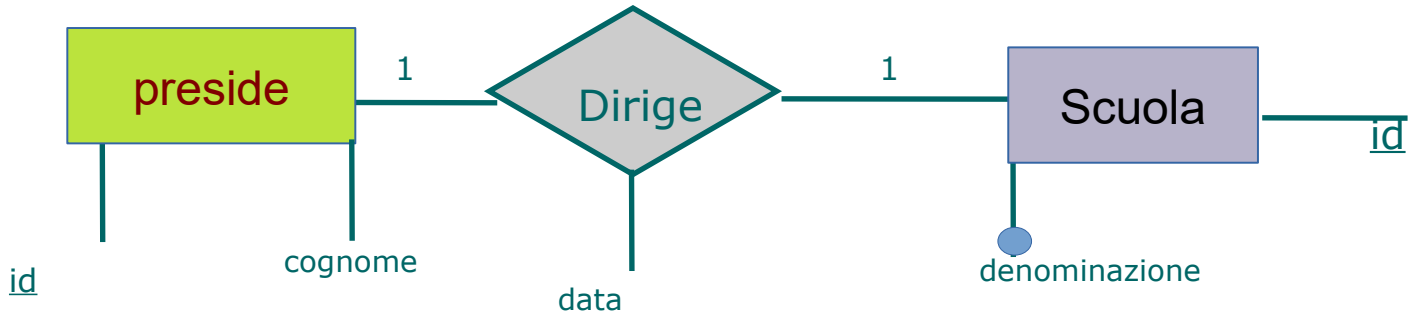


Istanza concettuale

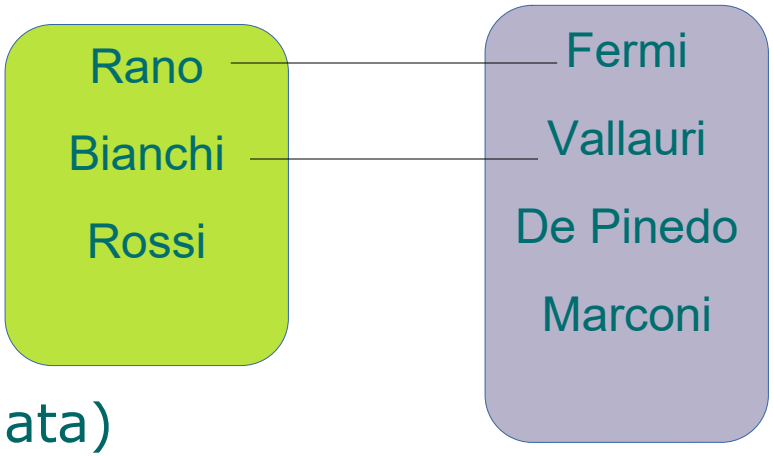


Determinare lo schema logico

Relazioni uno ad uno (esempio)



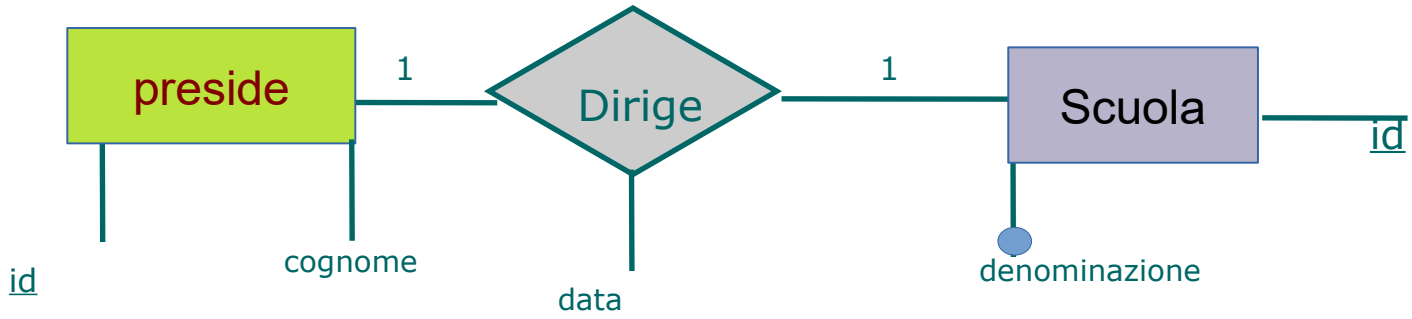
Istanza concettuale



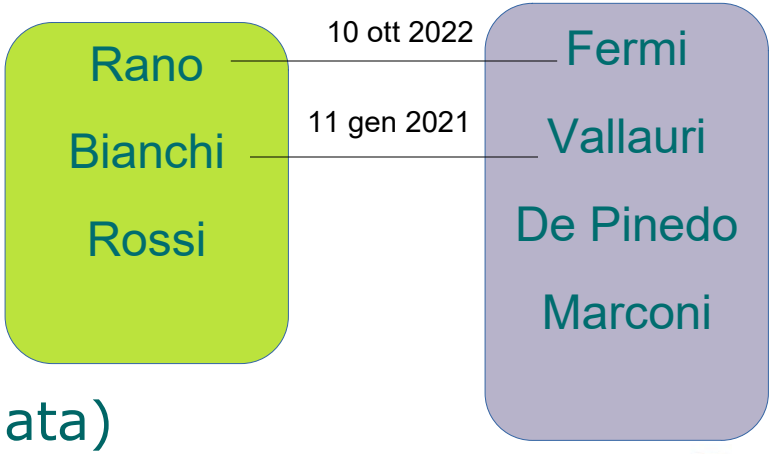
Scuola(id, denominazione^{u},cognome,data)

schema logico

Esercizio



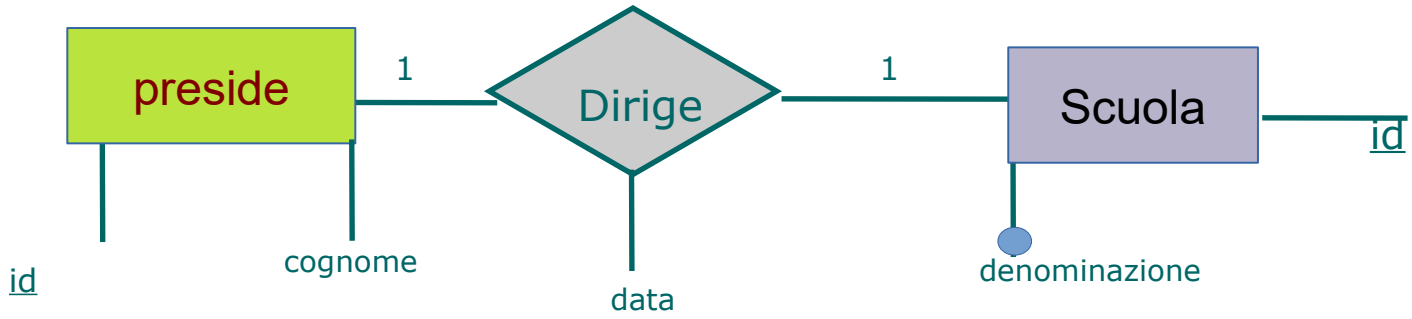
Popolare lo schema logico



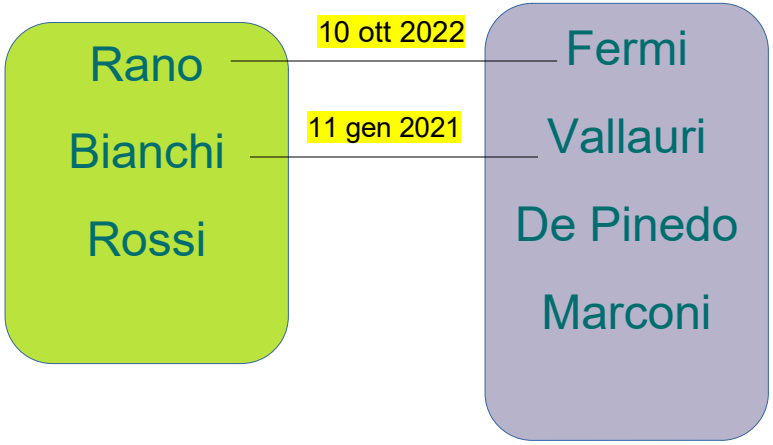
Scuola(id, denominazione^{u},cognome,data)

Istanza concettuale

Soluzione



Istanza concettuale



Istanza logica

id	denominazione	cognome	data
1	Fermi	Rano	10 ott 2022
2	Vallauri	Bianchi	11 gen 2021
3	De Pinedo	null	null
4	Marconi	null	null
5	null	Rossi	null

Scuola(id, denominazione^{u},cognome,data)

Step di conversione

1) Aggiungere Id a tutte le entità
2) Regola per entità (una tabella per ogni entità)
3) Regola per attributi (aka copiare attributi)
4) Regola per le relazioni (partendo da quelle periferiche e di ordine più basso)
5) *checklist*
6) Implementazione SQL

Regole per i vincoli:checklist

- Definire il campo "id" come CHIAVE PRIMARIA e AUTO_INCREMENT
- Implementare i vincoli di integrità referenziale
- Implementare i vincoli di tupla (dominio, range, formato)
- Tutti gli attributi identificatori (singoli o multipli) dovranno avere il vincolo di unicità (vincolo UNIQUE)
- Gli attributi ad esistenza obbligatoria "*" dovranno avere, a livello fisico, il vincolo "NOT NULL"
- Gli attributi identificatori e ad esistenza obbligatoria dovranno essere non nulli e unici
- Per le colonne FK (che non esistevano a livello concettuale) ragionare su
  - Unicità (vincolo di chiave, solo in rari casi)
  - Esistenza obbligatoria (NOT NULL)
    - Relazione 1→M (abitanti-RES-città):In caso di partecipazione obbligatoria la FK (lato M) dovrà avere il vincolo "NOT NULL" (ex `id_citta_res*`)
    - Relazione M→M: le FK dovranno avere sempre il vincolo "NOT NULL"
- Le cardinalità massime diverse da *n* vanno implementate lato codice
  - Ex turisti—PREN—viaggi prima di aggiungere una prenotazione devo controllare di non aver già raggiunto il massimo di posti disponibili

Esercizio: residenza e lavoro

SOLUZIONE:

CITTA(id, nome*, kmq)

ABITANTI(id, nome*, cognome*,id_città_residenza*,id_città_lavoro,stipendio,indirizzo*)

ID	NOME	KMQ
1	Roma	120
2	Milano	80
5	Ariccia	20

ID	NOME	COGNOME	ID_CITTA'_RES	ID_CITTA'_LAV	STIPENDIO	INDIRIZZO
1	Mario	Rossi	1	2	1300	Via Verdi
2	Gino	Bianchi	2	2	1500	Parco Sempione
3	Nino	Bianchi	2	1	1200	Navigli
4	Rino	Rano	1	null	null	Via Nazionale

Soluzione: Popolare lo schema logico

id	nome	cognome	telefono
1	rino	rano	338
2	gino	rossi	333
3	gina	rossi	null
4	mario	bianchi	335

id	nome	CF
1	Analisi1	5
2	fisica1	10
3	canto	5

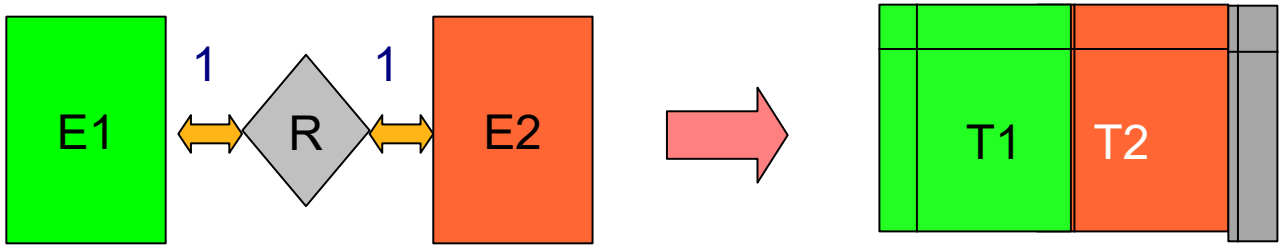
id	id_stud	id_mat	voto
1	1	1	18
2	1	2	20
3	3	2	24
4	3	3	25
5	1	3	30
6	4	3	18

Sintesi

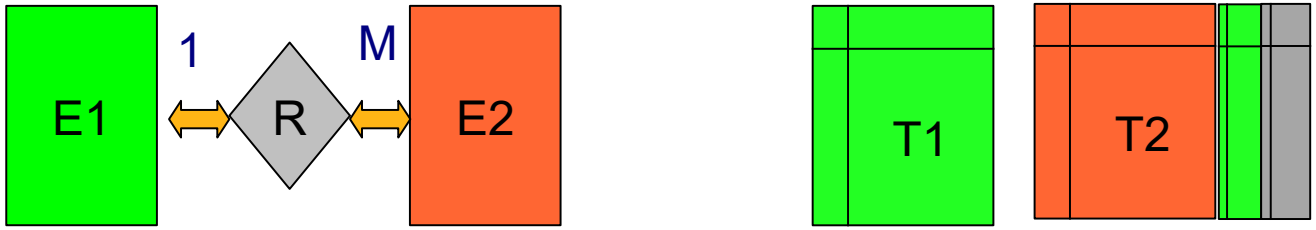
- **Entità** → relazioni (tabelle)
- **Attributi di entità** → attributi delle relazioni (colonne delle tabelle)
- Relazioni **1 a 1** → monotabella con un solo id e tutti gli attributi
- Relazioni **1 a M** → aggiungere al lato **M** la FK che si riferisce alla chiave primaria del lato **1**
- Relazioni **M a M** → creazione di una nuova tabella avente due FK che si riferiscono alle due chiavi primarie delle due Entità coinvolte nella Relazione

Sintesi Grafica

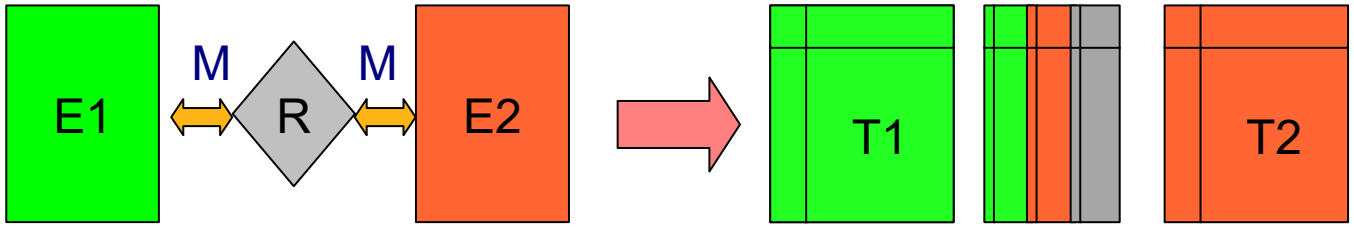
Uno a Uno



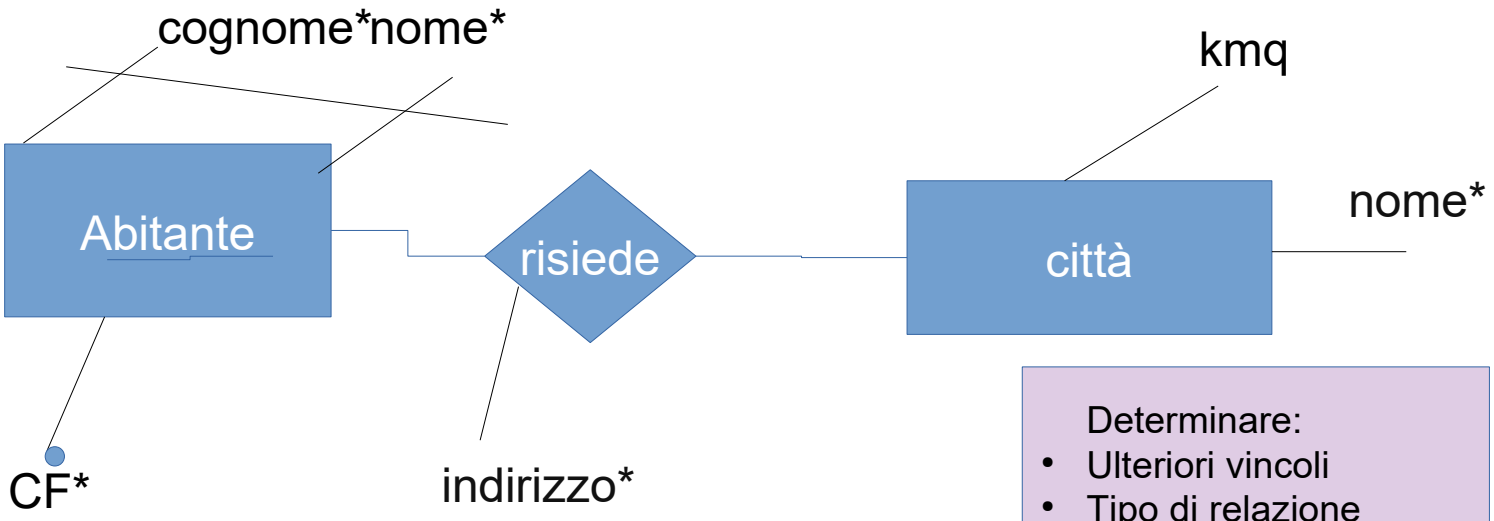
Uno a Molti



Molti a Molti



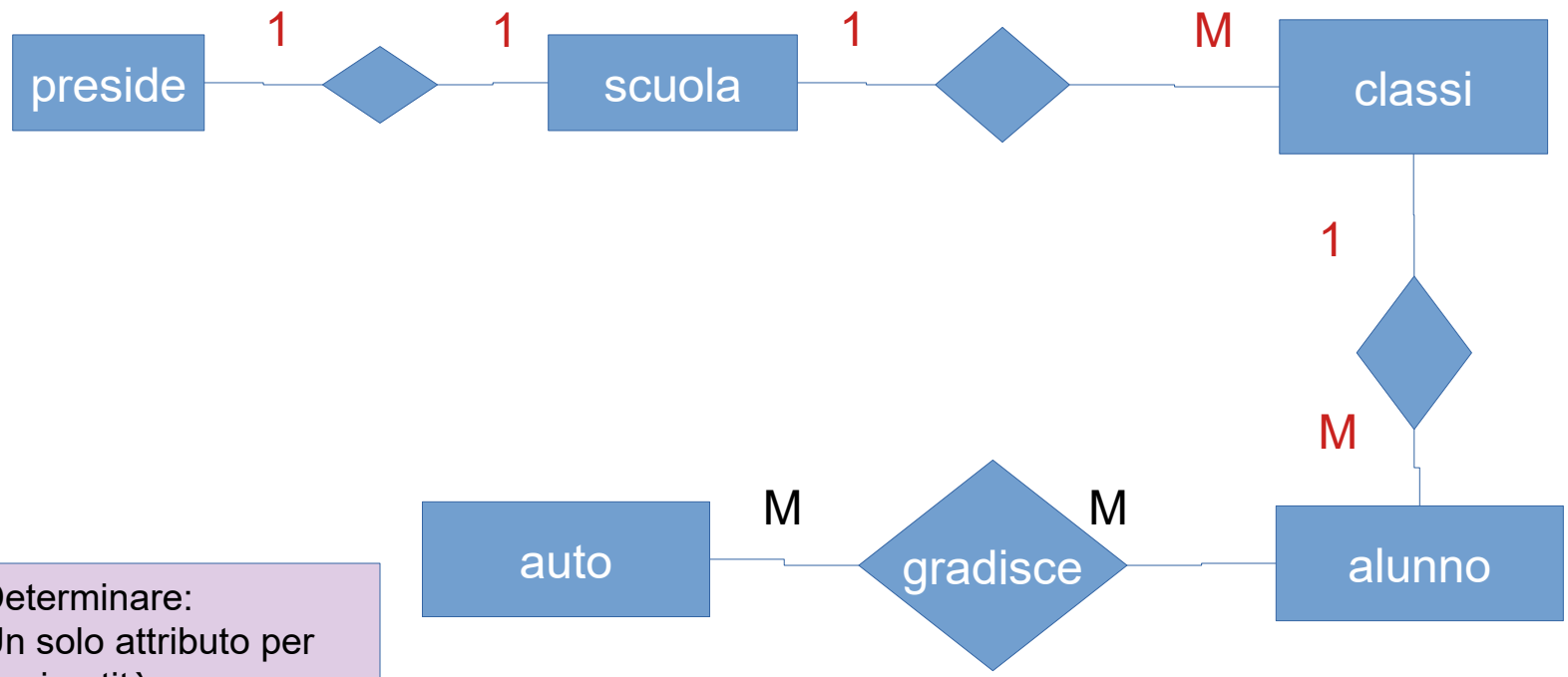
Esercizio: Residenza



Determinare:

- Ulteriori vincoli
- Tipo di relazione
- Schema logico

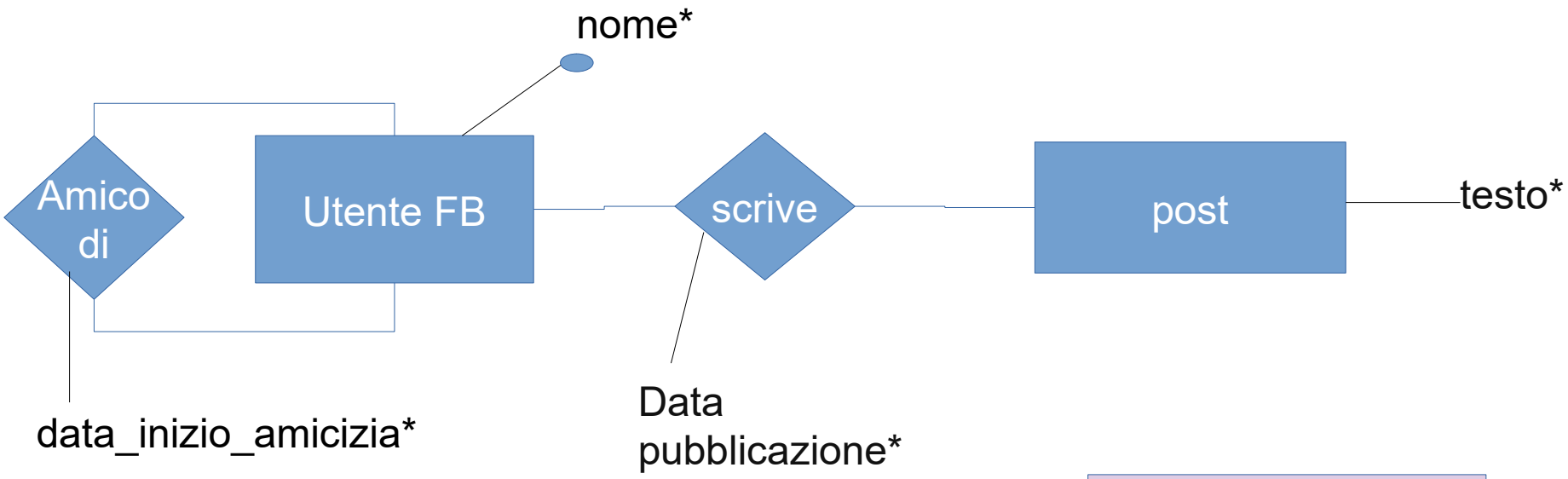
Esercizio Scuola



Determinare:

- Un solo attributo per ogni entità
- Ulteriori vincoli
- Nomi relazioni
- Tipo di relazione
- Schema logico

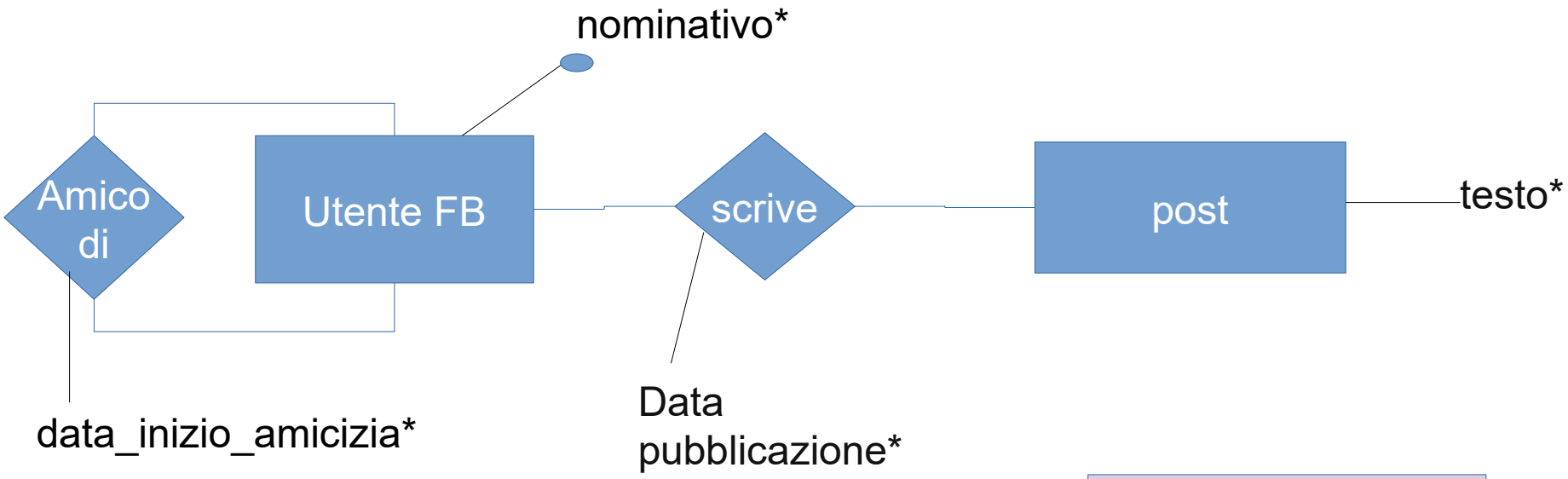
Esercizio: Facebook



Determinare:

- Ulteriori vincoli
- Tipo di relazione
- Schema logico

Esercizio: Facebook



Determinare:

- Ulteriori vincoli
- Tipo di relazione
- Schema logico

Facebook={

UTENTI(id,nome),

AMICIZIA(id, id_utente, id_amico, data),

POST(id, testo, id_utente, data_pubblicazione)

}

Esercizio facebook: schema logico

Facebook={
UTENTI(id,nome),
AMICIZIA(id, id_utente, id_amico, data),
POST(id, testo, id_utente
}

UTENTI

id	nome

AMICIZIA

id	id_utente	id_amico	data

POST

id	testo	id_utente

Esercizio: Istanza

UTENTI

id	nome
1	rino
2	gino
3	gina
4	mario

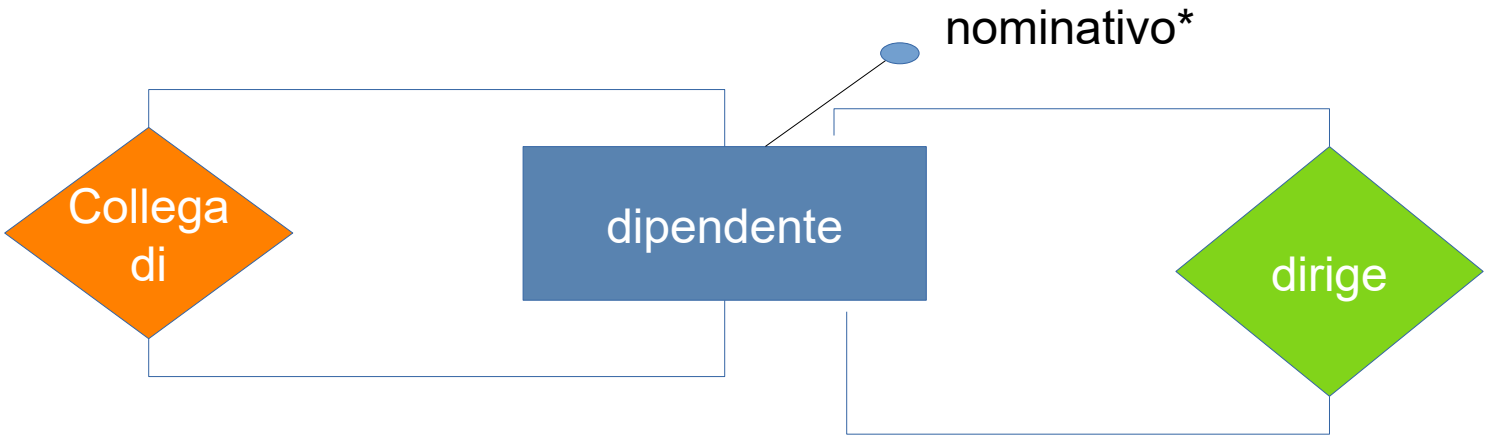
POST

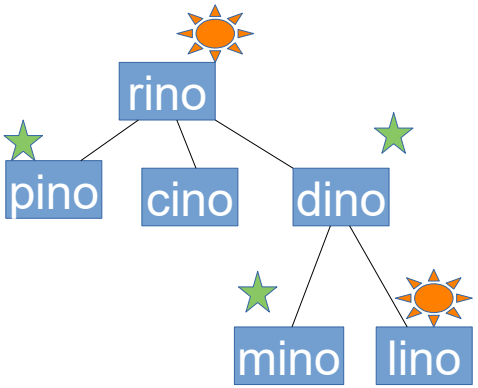
id	testo	id_utente
1	Il Java	1
2	Offerta lavoro	2
3	Buongiornissimo	1

AMICIZIA

id	id_ut	id_amico	data
1	1	2	2012/12/12
2	1	3	2013/02/12
3	2	4	2022/01/31

Esercizio: Azienda





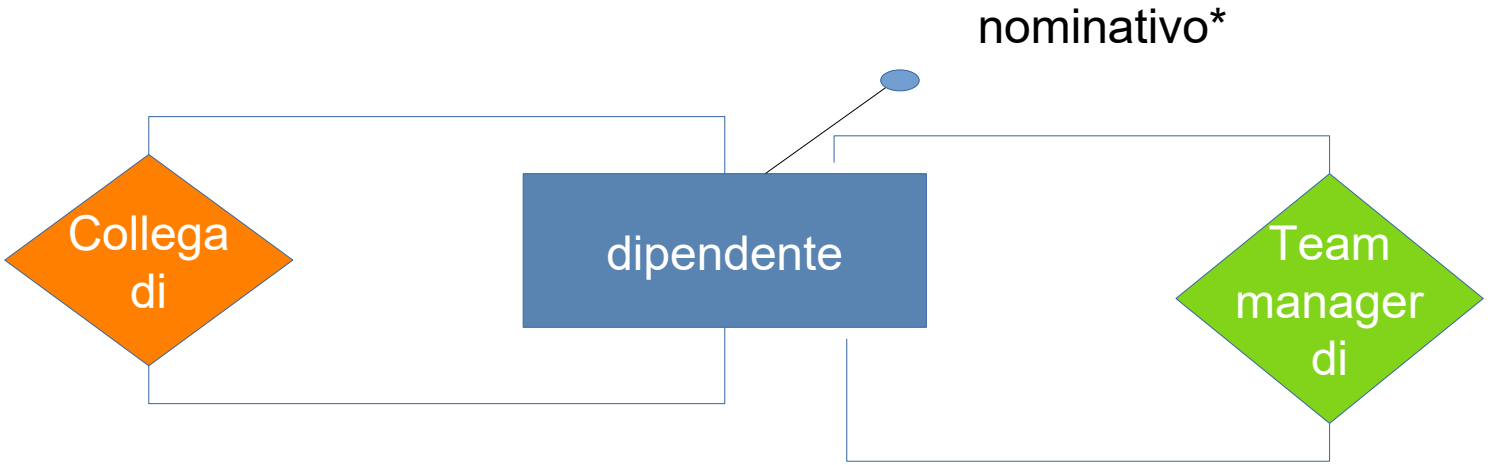
Rino e Lino sono colleghi

Pino, Dino e Mino sono colleghi

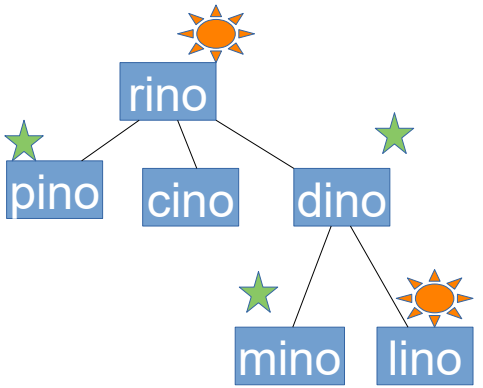
Determinare:

- Ulteriori vincoli
- Tipo di relazione
- Cardinalità min/max
- Schema logico
- Istanza (popolare le tabelle)

Esercizio Azienda: schema logico



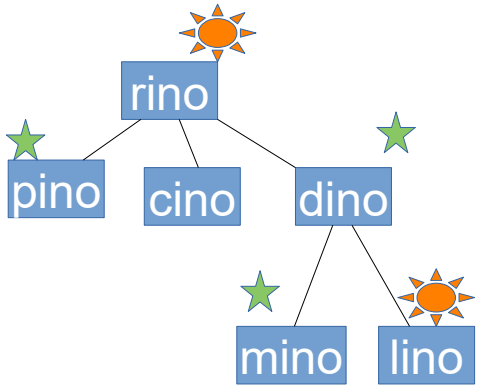
Azienda={
DIPENDENTI(id,nome, id_manager),
COLLEGHI(id, id_dipendente, id_collega)
}



Esercizio Azienda: istanza logica

DIPENDENTI

id	nome	id_manager
1	rino	null
2	pino	1
3	cino	1
4	dino	1
5	mino	4
6	lino	4



COLLEGHI

id	id_dipendente	id_collega
1	1	6
2	2	4
3	2	5
4	4	5