UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Dipartimento di Scienze Fisiche,
Informatiche e Matematiche

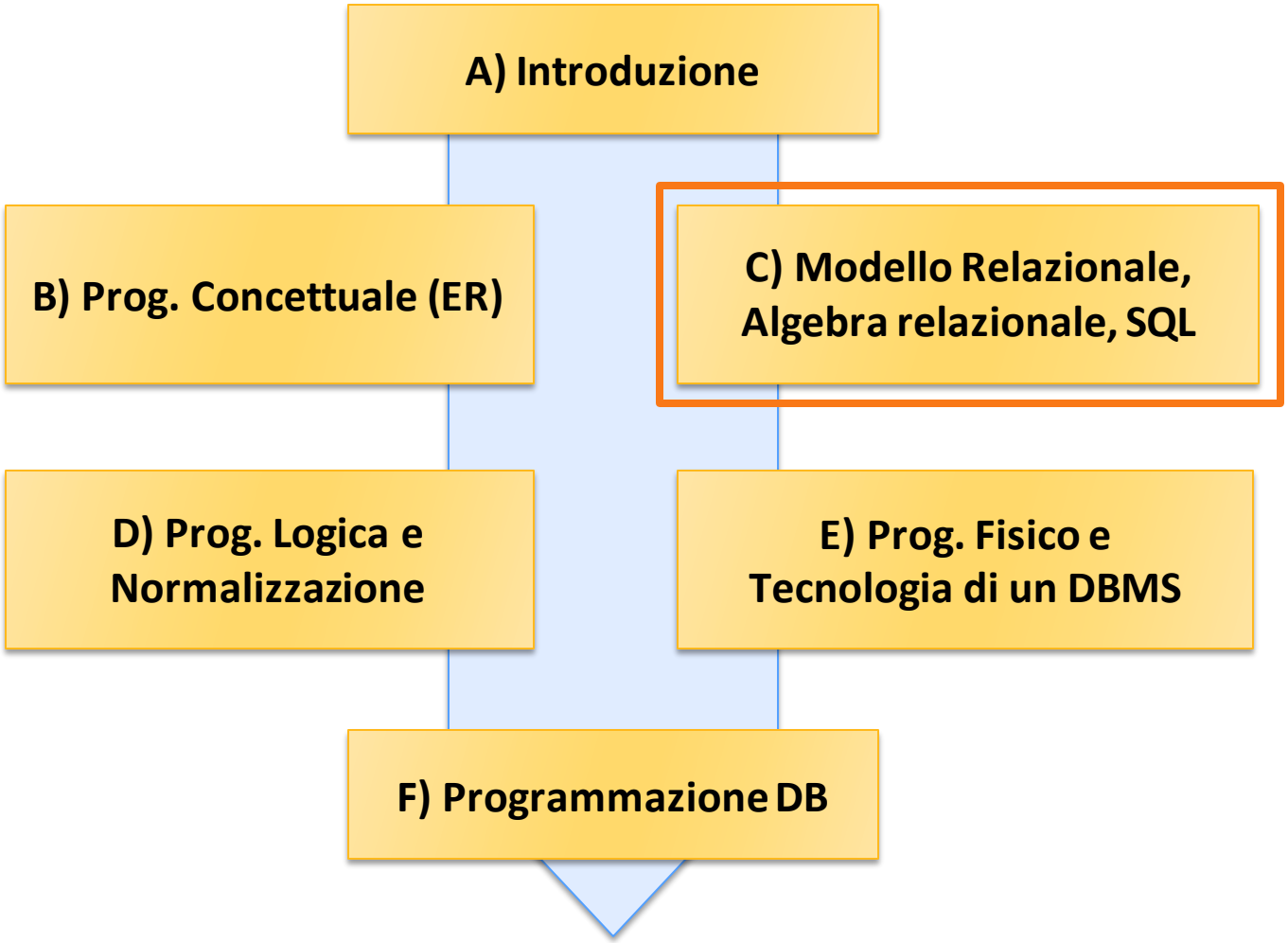
Basi di Dati

Corso di Laurea in Informatica

Basi di Dati

Dichiarazione degli schemi e interrogazioni in SQL (II Parte)

Basi di Dati – Dove ci troviamo?



Dichiarazione degli schemi

- Analizzeremo ora in dettaglio il problema dell'integrità referenziale (foreign key) e la sua gestione in SQL

Integrità referenziale

- Esprime un legame gerarchico (padre-figlio) fra tabelle
- Alcuni attributi della tabella figlio sono definiti FOREIGN KEY
- I valori contenuti nella FOREIGN KEY devono essere sempre presenti nella tabella padre

Integrità referenziale

- Espressa come vincolo di tabella:

  FOREIGN KEY (MATR) REFERENCES STUDENTI (MATR)

- Espressa come vincolo di colonna:

  MATR CHAR(6) REFERENCES STUDENTI (MATR)

Nota: in caso si ometta la lista delle colonne riferite, viene utilizzata la chiave primaria della tabella riferita. Ad esempio, il seguente comando è equivalente al primo:

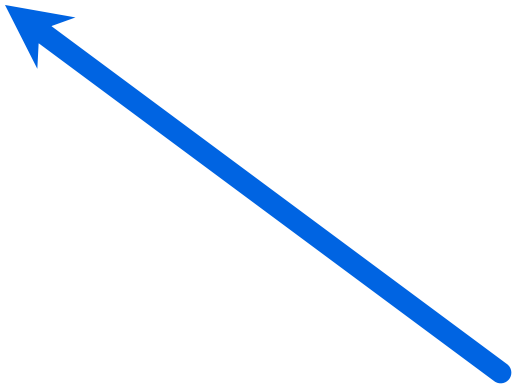
FOREIGN KEY (MATR) REFERENCES STUDENTI

Esempio: studente - esame

studente

MAT	
R	
123	
415	

702



esame

MAT	
R	
123	
123	

702

Il problema degli orfani

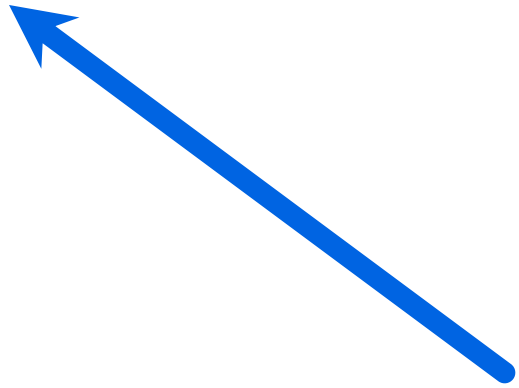
studente

MAT	
~~R~~	
123	
415	

702

ORFANI:

tuple che restano prive di padre a causa di cancellazioni
e modifiche della tabella padre



esame

MAT	
~~R~~	
~~123~~	
123	

702

Gestione degli orfani

- Cosa succede degli esami se si cancellano gli studenti?
  - CASCADE (si cancellano anche gli esami)
  - SET NULL
  - SET DEFAULT
  - NO ACTION

- Cose succede degli esami se si modifica la matricola nella tabella STUDENTE?
  - CASCADE (si modificano anche gli esami)
  - SET NULL
  - SET DEFAULT
  - NO ACTION

Gestione degli orfani

- CASCADE
  - Aggiornamenti su colonne riferite aggiornano tutte le colonne delle tuple con foreign key che si riferiscono ad esse
  - Cancellazioni di tuple riferite cancellano tutte le tuple contenenti riferimenti ad esse
- SET NULL
  - Aggiornamenti e cancellazioni su colonne riferite causano la modifica delle colonne di foreign key a NULL

Gestione degli orfani

- SET DEFAULT
  - Aggiornamenti e cancellazioni su colonne riferite causano la modifica delle colonne di foreign key al valore di default
- NO ACTION
  - Aggiornamenti e cancellazioni su colonne riferite sono proibiti se riferiti da almeno una tupla con foreign key

Definizione : nella tabella figlia

```
CREATE TABLE ESAME
   (       ……..
           FOREIGN KEY (MATR) REFERENCES STUDENTI
                   ON DELETE CASCADE ON UPDATE CASCADE )
```

- È lecito essere figli di più padri

```
CREATE TABLE ESAME
   (       ….
           PRIMARY KEY(MATR,COD-CORSO) ,
           FOREIGN KEY (MATR)  REFERENCES STUDENTI
                   ON DELETE CASCADE  ON UPDATE CASCADE,
           FOREIGN KEY (COD-CORSO) REFERENCES CORSO
                   ON DELETE NO ACTION ON UPDATE NO ACTION )
```

Una istanza scorretta

studente

MAT	NOME	CITT	C-DIP
R		À	
123			
415			
702			

esame

MATR	COD-CORSO	DATA	VOTO
	1	2014-09-07	
123	2	2015-01-08	30
123	2	2014-08-01	28
123	2	2014-09-07	28
702	1	NULL	20
702	1	2014-09-07	NULL
714			28

viola la chiave

viola il NOT NULL

viola la integrità referenziale

Una istanza corretta

studente

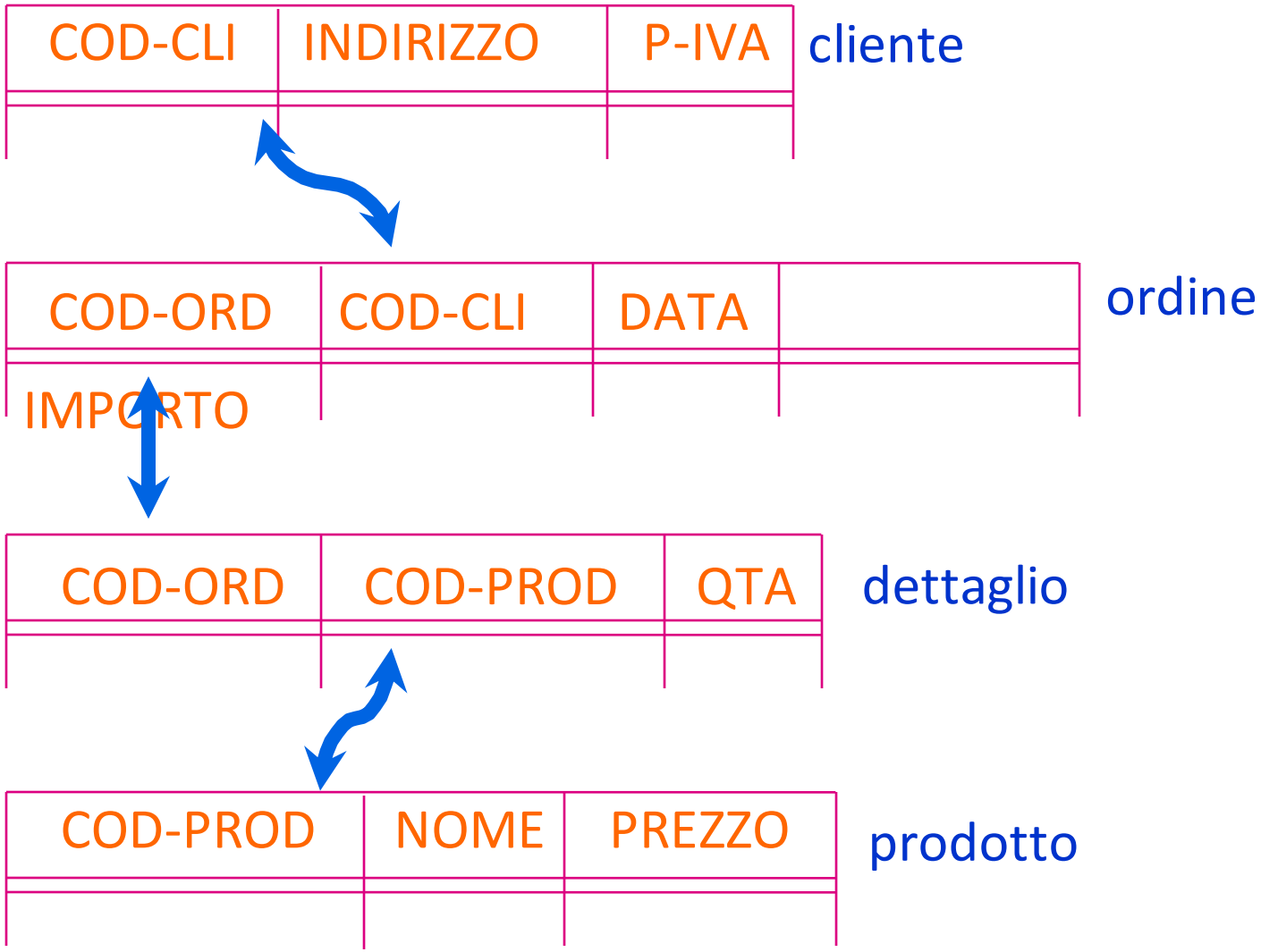
MAT	NOME	CITT	C-DIP
R		À	
123			
415			

702

esame

MAT R	COD-CORSO	DATA	VOT O
	1	2014-09-07	
123	2	2015-01-08	30
123	2	2014-09-07	28
702			20

Esempio : gestione ordini

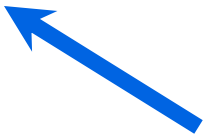


Definizione della tabella CLIENTE

cliente

COD-CLI	INDIRIZZO	P-IVA

```
CREATE TABLE CLIENTE
   (      COD-CLI  CHAR(6)  PRIMARY KEY,
          INDIRIZZO  CHAR(50),
          P-IVA CHAR(12) NOT NULL UNIQUE
   )
```



Chiave alternativa

Definizione della tabella ORDINE

ordine

COD-ORD	COD-CLI	DATA	IMPORTO

```
CREATE TABLE ORDINE
   (       COD-ORD  CHAR(6)  PRIMARY KEY,
           COD-CLI CHAR(6)  NOT NULL  DEFAULT='999999',
           DATA  DATE,
           IMPORTO  INTEGER,
           FOREIGN KEY (COD-CLI) REFERENCES CLIENTE
                   ON DELETE SET DEFAULT
                   ON UPDATE SET DEFAULT
   )
```

Definizione della tabella DETTAGLIO

dettaglio

COD-ORD	COD-PROD	QTA

```
CREATE TABLE DETTAGLIO
   (        COD-ORD  CHAR(6),
            COD-PROD CHAR(6),
            QTA  SMALLINT,
            PRIMARY KEY (COD-ORD,COD-PROD),
            FOREIGN KEY (COD-ORD) REFERENCES ORDINE
                        ON DELETE CASCADE   ON UPDATE CASCADE,
            FOREIGN KEY (COD-PROD) REFERENCES PRODOTTO
                        ON DELETE NO ACTION   ON UPDATE NO ACTION
   )
```

Definizione della tabella PRODOTTO

prodotto

COD-PROD	NOME	PREZZO

```
CREATE TABLE PRODOTTO
  (     COD-PROD  CHAR(6)  PRIMARY KEY,
        NOME  CHAR(20),
        PREZZO  SMALLINT
  )
```

Esercizio: gestione personale

- esprimere in SQL la dichiarazione dello schema

impiegato

MAT	NOME	DATA-ASS	SALARIO	MATR-MGR
R	Piero	2012-01-01	1500 €	
1	Giorgio	2014-01-01	2000 €	2
2	Giovanni	2013-07-01	1000 €	null

assegnamento

MAT	NUM-PROG	PERC
R	3	50
1	4	50
1	3	100
2	4	100

progetto

NUM-PROG	TITOL	TIPO
3	O	Esprit
4	Idea	Esprit

Interrogazioni in SQL

- Continueremo ora con l'analisi di come realizzare interrogazioni di base in SQL

Sintassi nella clausola SELECT

- SELECT *
- SELECT NOME, CITTÀ
- SELECT DISTINCT CITTÀ
- SELECT CITTÀ AS LUOGO-DI-RESIDENZA
- SELECT REDDITO-CATASTALE * 0.05
  AS TASSA-ICI
- SELECT SUM (SALARIO)

Sintassi della clausola FROM

- FROM STUDENTE
- FROM STUDENTE AS X
- FROM STUDENTE X
- FROM STUDENTE, ESAME
- FROM STUDENTE JOIN ESAME
  ON STUDENTE.MATR=ESAME.MATR

Sintassi del predicato di selezione

Espressione booleana di predicati semplici

- operazioni booleane :
  - AND (P1 AND P2)
  - OR (P1 OR P2)
  - NOT (NOT P1)
- predicati semplici :
  - TRUE, FALSE
  - termine comparatore termine
- comparatore :
  - =, !=, <, <=, >, >=
- termine :
  - costante, attributo
  - espressione aritmetica di costanti e attributi

Sintassi della clausola WHERE

- Espressione Booleana di predicati semplici (come in algebra)

- Alcuni predicati aggiuntivi:
  - BETWEEN:
    - DATA BETWEEN 2007-01-01 AND 2016-12-31
  - LIKE:
    - C-DIP LIKE 'log%' → stringa arbitraria
    - TARGA LIKE 'E_777CX' → carattere arbitrario

Sintassi della clausola WHERE

- WHERE NOME LIKE ‘B%’
  - BOFFI
  - BUCCHI
  - BIANCHI
  - BIFFI
  - BONFATTI
- WHERE NOME LIKE ‘BI%’
  - BIANCHI
  - BIFFI
- Operatori aritmetici nel WHERE:
  - WHERE SALARIO + STRAORD >18
  - WHERE STRAORD + 5 > SALARIO

Gestione duplicati (proiezione)

SELECT DISTINCT C-DIP
FROM STUDENTE

C-DIP
Inf
Log

SELECT C-DIP
FROM STUDENTE

C-DIP
Inf
Inf
Log

Valori nulli

SELECT *

FROM STUDENTE

WHERE CITTÀ IS [NOT] NULL

- Attenzione :
  - se CITTÀ ha valore NULL il risultato per (CITTÀ = 'Milano') ha valore 'UNKNOWN'

Composizione di predicati con valore nullo

Logica a tre valori (V,F,U)

(Vero, Falso, Unknown)

V AND U = U

V OR U = V

F AND U = F

F OR U = U

NOT U = U

P = (CITTÀ IS NOT NULL) AND (C-DIP LIKE '%Inf')

CITTA'	C-DIP
Milano	Inf
Milano	NULL
NULL	Inf
Milano	Log

P	TUPLA SELEZ.
V	si
U	no
F	no
F	no

Join di due tabelle

```
SELECT NOME
FROM STUDENTE, ESAME
WHERE  STUDENTE.MATR = ESAME.MATR
AND C-DIP LIKE 'In%' AND VOTO = 30
```

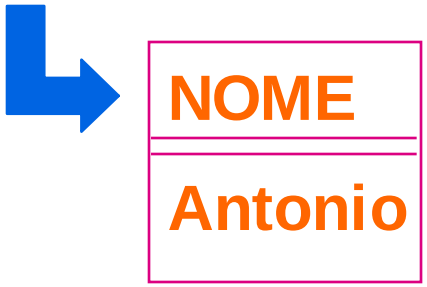
NOME
Carlo

Variante sintattica:

```
SELECT NOME
FROM STUDENTE JOIN ESAME
    ON STUDENTE.MATR = ESAME.MATR
WHERE C-DIP LIKE 'In%' AND VOTO = 30
```

Join di tre tabelle

```
SELECT NOME
FROM STUDENTE, ESAME, CORSO
WHERE  STUDENTE.MATR = ESAME.MATR
  AND CORSO.COD-CORSO = ESAME.COD-CORSO
  AND TITOLO LIKE 'info%' AND VOTO < 24
```



NOME
Antonio

Interrogazioni con variabili relazionali

- Es: chi sono i dipendenti “non-pendolari”?

impiegato

MATR	NOME	CITTÀ	SALARIO	MATR-MGR
1	Piero	BO	1500 €	2
2	Giorgio	MO	2000 €	4
3	Giovanni	FE	1000 €	2
….	…..	….	….	…

dipartimento

DNO	NOME	CITTÀ
1	AMMINISTRAZIONE	BO
2	SPEDIZIONI	FE
…	…..	…

assegnamento

MAT	DNO
R	1
1	1
3	…

Variabili relazionali

```
SELECT  I.NOME
FROM IMPIEGATO AS I, DIPARTIMENTO AS D,
     ASSEGNAMENTO AS A
WHERE  I.MATR = A.MATR
AND D.DNO = A.DNO
AND I.CITTÀ = D.CITTÀ
```

(NON PENDOLARI)

AND I.CITTÀ != D.CITTÀ

(PENDOLARI)

Variabili relazionali (self-join)

- Es: Chi sono i dipendenti di Giorgio?

```
SELECT  X.NOME
FROM IMPIEGATO AS X, IMPIEGATO AS Y
WHERE  X.MATR-MGR = Y.MATR
   AND Y.NOME = 'Giorgio'
```

X.NOME
Piero
Giovanni

Esercizi

- Dato un DB per la gestione del personale esprimere in SQL le interrogazioni seguenti:
  - in quali tipi di progetti lavora Giovanni?
  - chi è il manager di Piero?
  - in quali progetti lavora Piero?
  - quali impiegati lavorano nel progetto “IDEA”?
  - quali impiegati lavorano al 100% del loro tempo nel progetto “WIDE”?

- E le modifiche:
  - inserire la tupla <4,Luca,2014-01-01,2M,1>
  - modificare il salario di Piero in 3000€
  - aumentare il salario di Giorgio del 5%
  - cancellare i dati di Giovanni

Esempio : gestione personale

impiegato

MATR	NOME	DATA-ASS	SALARIO	MATR-MGR
1	Piero	2012-01-01	1500 €	2
2	Giorgio	2014-01-01	2000 €	null
3	Giovanni	2013-07-01	1000 €	2

assegnamento

MATR	NUM-PROG	PERC
1	3	50
1	4	50
2	3	100
3	4	100

progetto

NUM-PROG	TITOLO	TIPO
3	Idea	Esprit
4	Wide	Esprit

Esercizi

- **in quali tipi di progetti lavora Giovanni?**

```
SELECT TIPO
FROM IMPIEGATO AS I,  ASSEGNAMENTO AS A,
          PROGETTO AS P
WHERE I.MATR=A.MATR
AND A.NUM-PROG=P.NUM-PROG
AND NOME='Giovanni'
```

TIPO
Esprit

- **chi è il manager di Piero?**

```
SELECT Y.NOME
FROM IMPIEGATO AS X, IMPIEGATO AS
Y
WHERE X.MATR-MGR=Y.MATR
AND X.NOME='Piero'
```

NOME
Giorgio

Esercizi

- **modificare il salario di Piero in 3000 €**

```
UPDATE IMPIEGATO
SET SALARIO = 3000
WHERE NOME='Piero'
```

- **aumentare il salario di Giorgio del 5%**

```
UPDATE IMPIEGATO
SET SALARIO = SALARIO * 1.05
WHERE NOME='Giorgio'
```