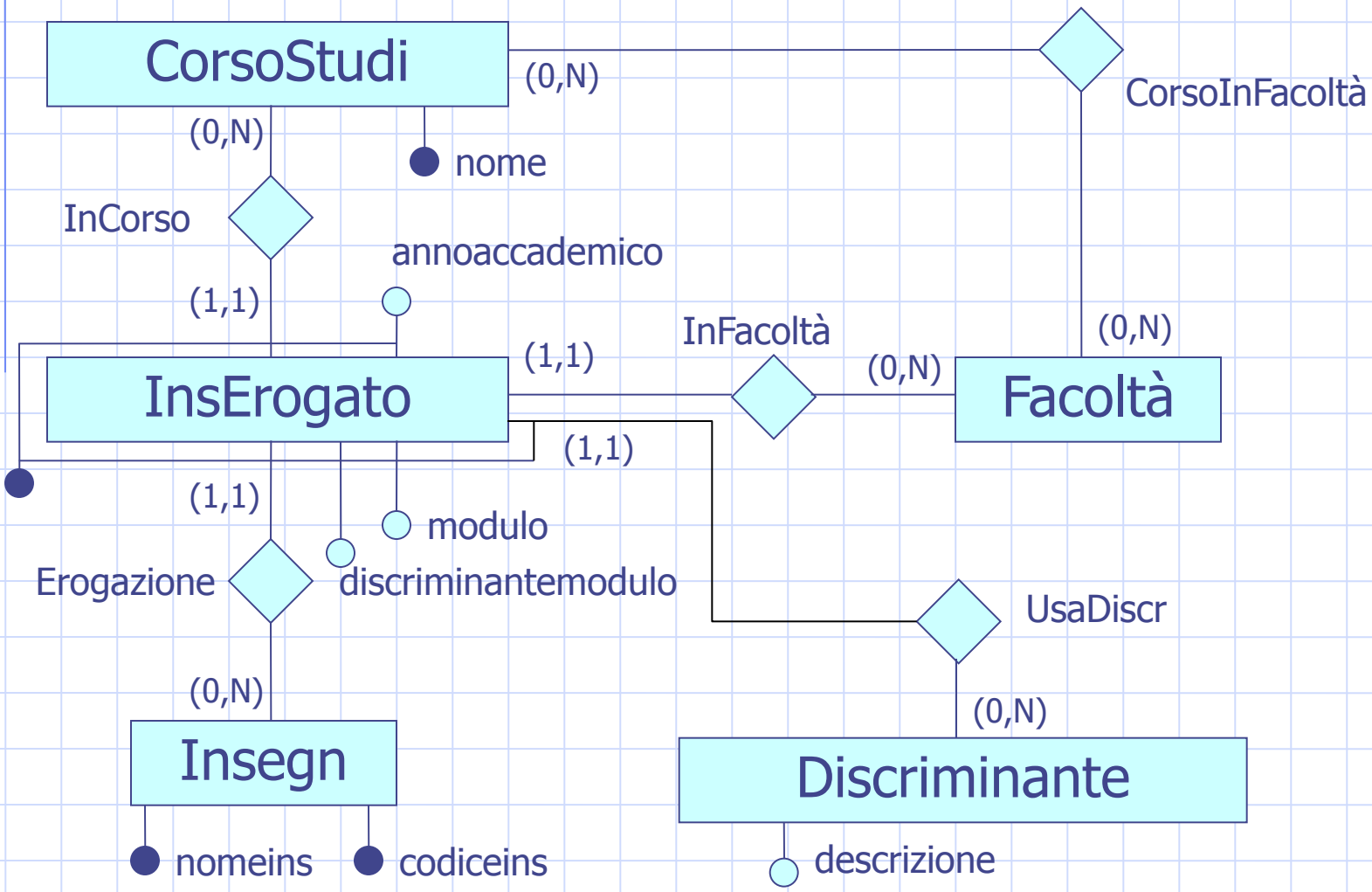
Laboratorio di Basi di dati & Basi di dati per Bioinformatica

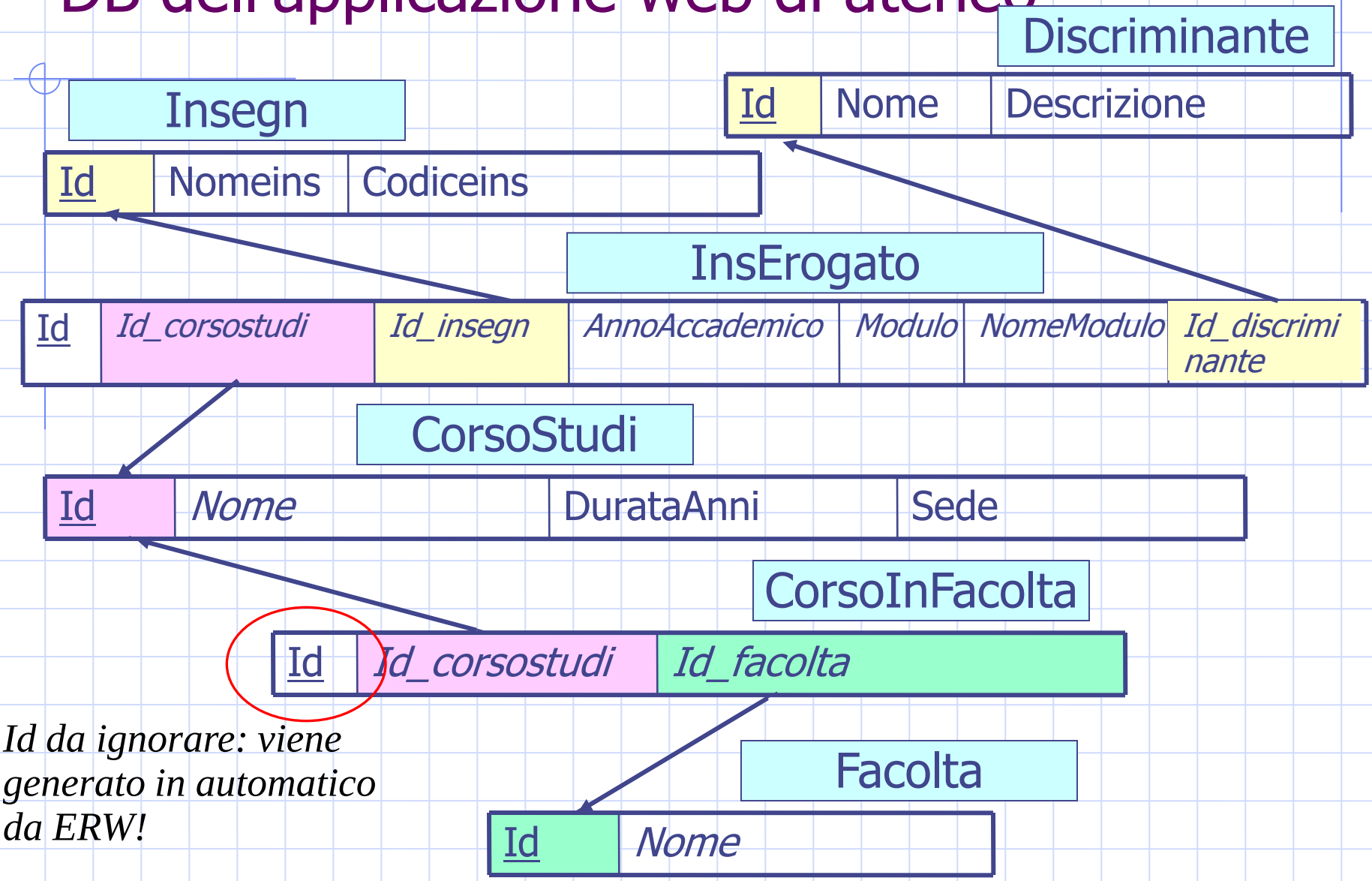
Docenti: Alberto Belussi & Carlo Combi

Lezione 4

Base di Dati usata negli esercizi (DB dell'applicazione web di ateneo)



DB dell'applicazione web di ateneo



Id da ignorare: viene generato in automatico da ERW!

Ulteriori esempi sugli insegnamenti erogati

Insegnamento non diviso in moduli:

- Programmazione (5 crediti)

InsErogato

Id	Anno accademico	Id_Corsostudi	Id_insegn	Id_discriminante	Modulo	Nome modulo	Crediti
2	2006/2007	1	98		0		5

Hamoduli	Mutuato	Id_facolta	Discriminantemodulo		
0	0	1			

Ulteriori esempi sugli insegnamenti erogati

Insegnamento diviso in moduli:

- Programmazione (10 crediti)
  - Modulo Teoria (8 crediti)
  - Modulo Laboratorio (2 crediti)

InsErogato

Id	Anno accademico	Id_corsostudi	Id_insegn	Id_discriminante	Modulo	Nome modulo	Crediti
1	2006/2007	1	98		0		10
2	2006/2007	1	98		1	Teoria	8
3	2006/2007	1	98		2	Laboratorio	2

Hamoduli	Mutuato	Id_facolta	Discriminantemodulo		
1	0	1			
0	0	1			
0	0	1			

Ulteriori esempi sugli insegnamenti erogati

Insegnamento diviso in moduli replicati:

- Programmazione (10 crediti)
  - Modulo Teoria (8 crediti)
  - Modulo Laboratorio [Sezione A] (2 crediti)
  - Modulo Laboratorio [Sezione B] (2 crediti)

InsErogato

Id	Anno accademico	Id_corsostudi	Id_insegn	Id_discriminante	Modulo	Nome modulo	Crediti
1	2006/2007	1	98		0		10
2	2006/2007	1	98		1	Teoria	8
3	2006/2007	1	98		2	Laboratorio	2
4	2006/2007	1	98		2	Laboratorio	2

Hamoduli	Mutuato	Id_facolta	Discriminantemodulo
1	0	1	
0	0	1	
0	0	1	Sezione A
0	0	1	Sezione B

Ulteriori esempi sugli insegnamenti erogati

Insegnamento replicato senza moduli:

- Programmazione [Sezione A] (10 crediti)
- Programmazione [Sezione B] (10 crediti)

InsErogato

Id	Anno accademico	Id_corsostudi	Id_insegn	Id_discriminante	Modulo	Nome modulo	Crediti
1	2006/2007	1	98	1	0		10
2	2006/2007	1	98	2	0		10

Hamoduli	Mutuato	Id_facolta	Discriminantemodulo		
0	0	1			
0	0	1			

Discriminante

Id	descrizione
1	Sezione A
2	Sezione B

Ulteriori esempi sugli insegnamenti erogati

Insegnamento replicato con unità logistiche:

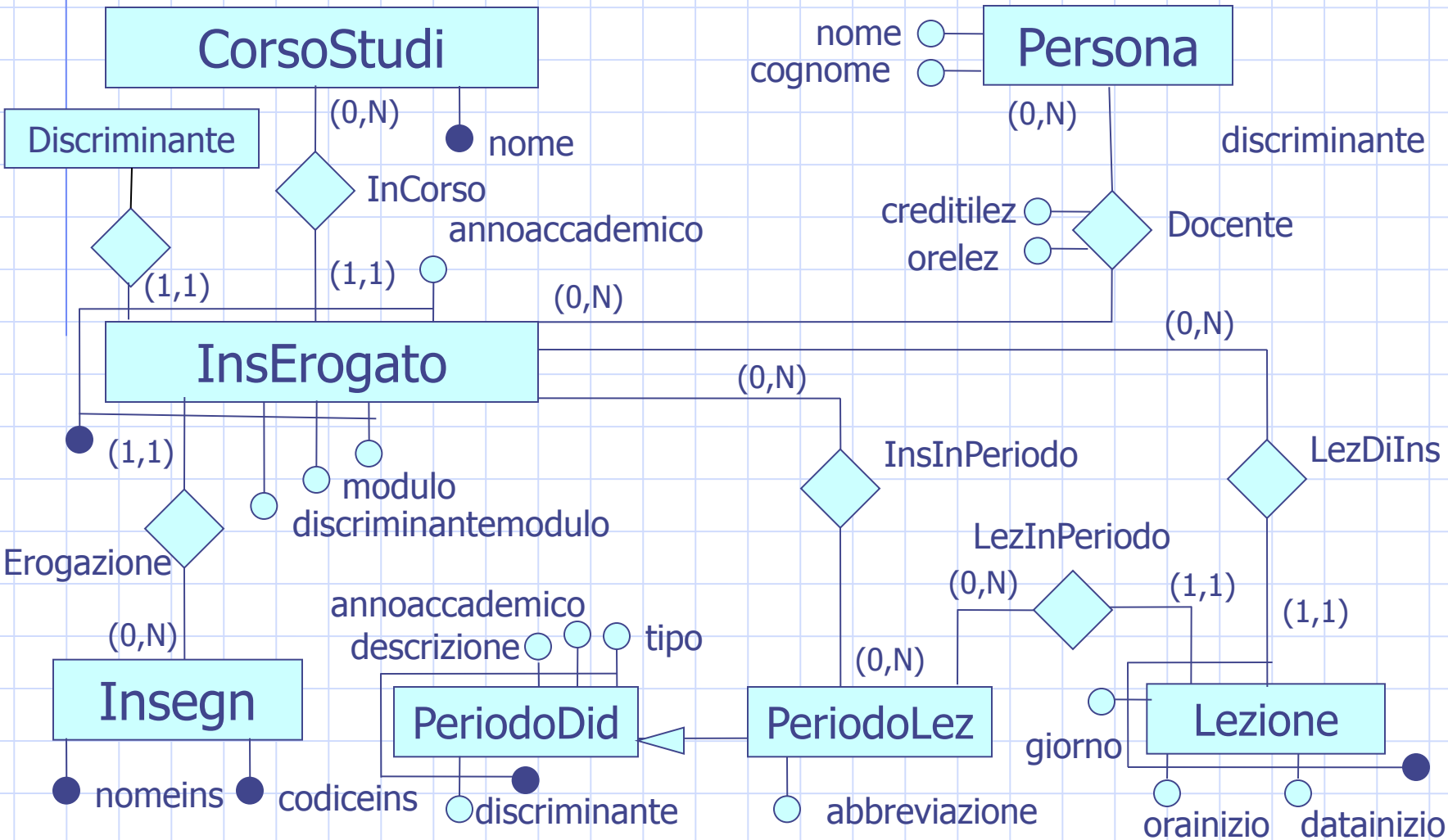
- Programmazione [Sezione A] (10 crediti)
  - Unità Teoria (8 crediti)
  - Unità Laboratorio 1 (1 crediti)
  - Unità Laboratorio 2 (1 crediti)

InsErogato

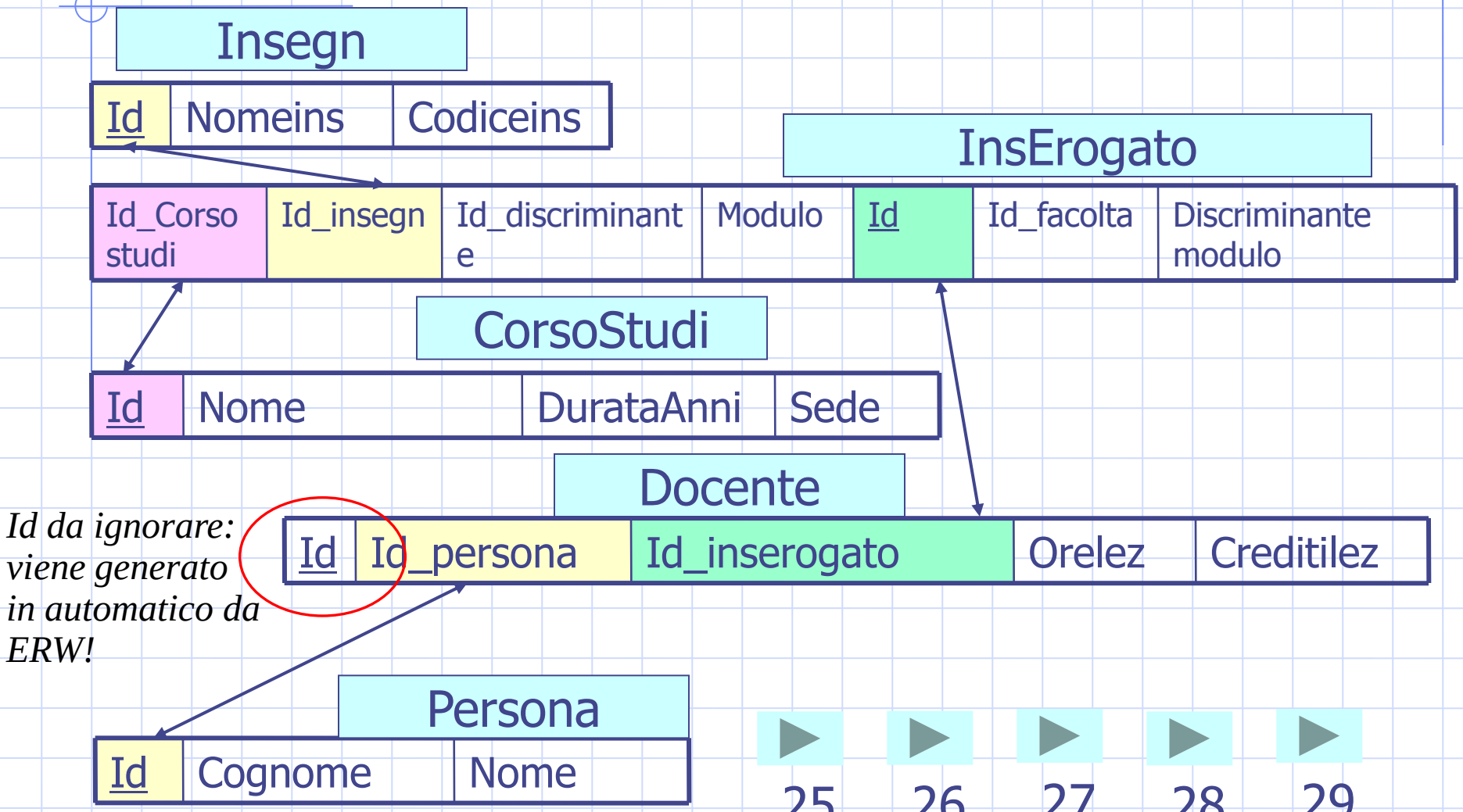
Id	Anno accademico	Id_corsostudi	Id_insegn	Id_discriminante	Modulo	Nome unità	Crediti
1	2010/2011	1	98	1	0		10
2	2010/2011	1	98	1	-1	Teoria	8
3	2010/2011	1	98	1	-2	Laboratorio 1	1
4	2010/2011	1	98	1	-2	Laboratorio 2	1

Haunità	Mutuato	Id_facolta	Discriminantemodulo		
1	0	1			
0	0	1			
0	0	1			
0	0	1			

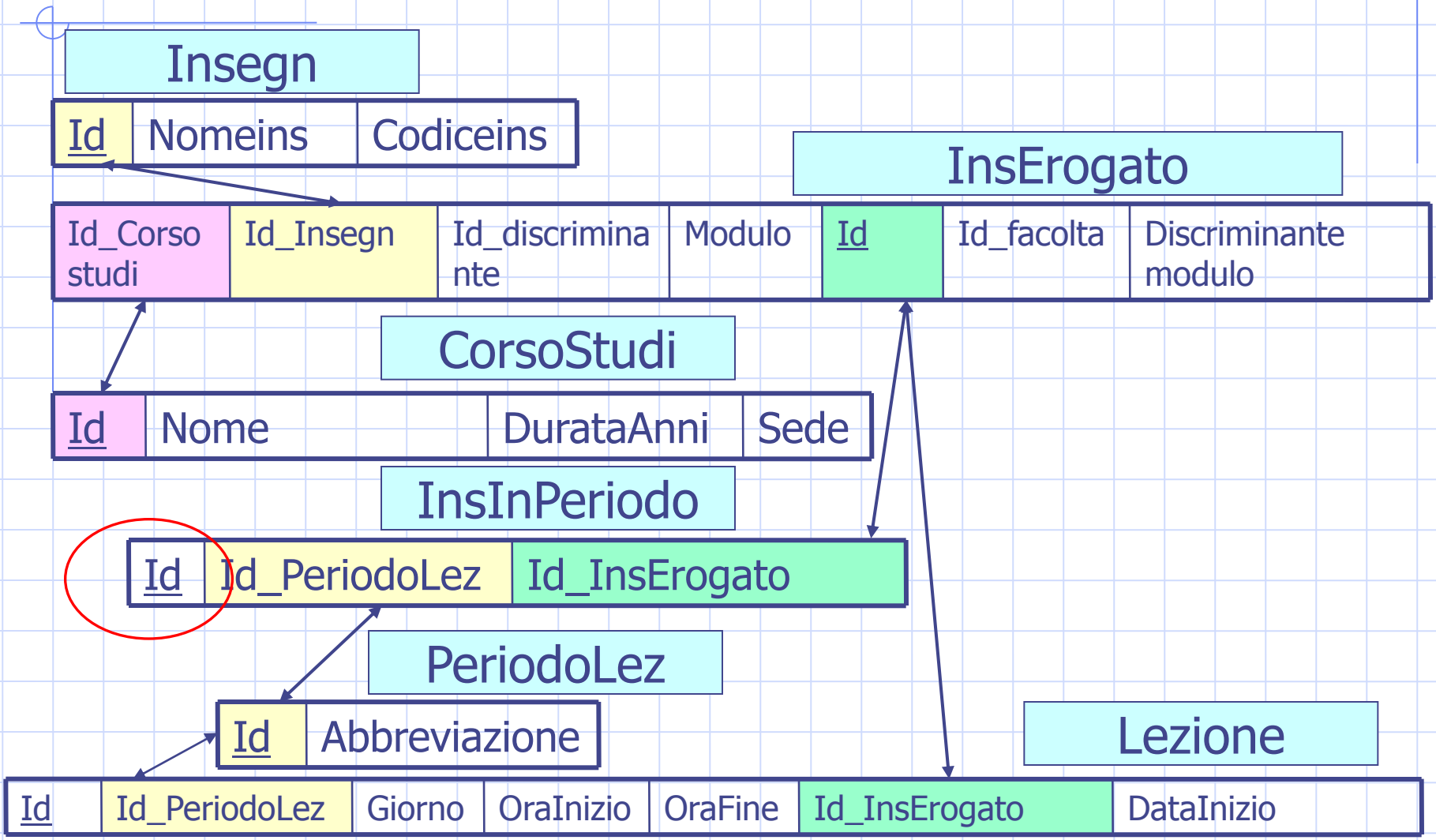
Base di Dati usata negli esercizi (DB dell'applicazione web di ateneo)



DB dell'applicazione web di ateneo



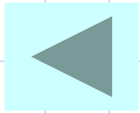
DB dell'applicazione web di ateneo



Esempio 25

- Visualizzare il nome degli insegnamenti offerti dal corso di laurea in informatica nell'anno accademico 2009/2010.

```
SELECT DISTINCT I.Nomeins
FROM   CorsoStudi CS, InsErogato IE, Insegn I
WHERE CS.id = IE.id_corsostudi
  AND IE.id_insegn = I.id
  AND OI.annoaccademico = '2009/2010'
  AND CS.nome = 'Informatica'
```



Esempio 26

- Visualizzare in ordine decrescente rispetto al cognome, tutti i docenti dell'ateneo coinvolti nella docenza di un insegnamento il cui nome contiene la sottostringa "Lingua", riportando nome e cognome del docente.

```
SELECT DISTINCT P.Nome, P.Cognome
FROM Persona P, Docente D, InsErogato IE, Insegn I
WHERE P.id = D.id_persona
  AND IE.id = D.id_inserogato
  AND IE.id_insegn = I.id
  AND I.nomeins like '%Lingua%'
ORDER BY Cognome DESC
```

Esempio 27

- Visualizzare, per ciascun corso di studi della Facoltà di Economia, il numero di docenti che hanno tenuto insegnamenti nel 2006/2007, riportando il nome del corso e il conteggio richiesto.

```
SELECT CS.nome, count(DISTINCT D.id_persona) AS NumDoc
FROM CorsoStudi CS, InsErogato IE, Docente D,
CorsoInFacolta CSF, Facolta F
WHERE CS.id = IE.id_corsostudi
  AND IE.id = D.id_inserogato
  AND CSF.id_corsostudi = CS.id
  AND CSF.id_facolta = F.id
  AND IE.annoaccademico = '2006/2007'
  AND F.nome = 'Economia'
GROUP BY CS.nome
```

Esempio 28

- Trovare i docenti che nel 2005/2006 hanno tenuto più di un insegnamento (o modulo/unità) riportando il numero degli insegnamenti tenuti, il numero totale di ore corrispondenti, il nome e il cognome del docente.

```
SELECT P.Nome, P.Cognome
  count(DISTINCT D.id_inserogato) AS NumIns,
  sum(D.orelez) AS OreTotaliDocenza
FROM InsErogato IE, Docente D, Persona P
WHERE EI.id = D.id_inserogato
  AND P.id = D.id_persona
  AND IE.annoaccademico = '2005/2006'
GROUP BY P.id, P.Nome, P.Cognome
HAVING count(DISTINCT D.id_inserogato) > 1
```

Join interni ed esterni

- SQL-2 ha introdotto una sintassi alternativa per l'espressione dei join che permette di distinguere le condizioni di join dalle condizioni di selezione sulle tuple.

```
SELECT AttrExpr [ [AS] Alias ]
        {, AttrExpr [ [AS] Alias ] }
FROM Tabella [ [AS] Alias ]
{ TipoJoin JOIN Tabella [ [AS] Alias ]
        ON CondizioneDiJoin }
[ WHERE CondizioneDiSelezione]
```

inner,
right outer,
left outer,
full outer

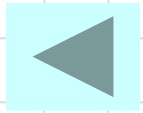
Join interni ed esterni

- INNER JOIN (JOIN INTERNO): rappresenta il tradizionale thetajoin dell'algebra relazionale. Con questo JOIN vengono selezionate solo le tuple del prodotto cartesiano per cui la condizione precisata nella clausola ON è vera.
- OUTER JOIN (JOIN ESTERNO): viene eseguito il JOIN mantenendo tutte le tuple che fanno parte di una o entrambe le tabelle.
  - LEFT JOIN: fornisce come risultato il join interno esteso con le tuple della relazione che compare a sinistra nel join per le quali non esiste una corrispondente tupla nella tabella di destra ("tuple escluse").
  - RIGHT JOIN: restituisce oltre al join interno, le "tuple escluse" della relazione di destra.
  - FULL JOIN: restituisce il join interno esteso con le "tuple escluse" di entrambe le relazioni.

Esempio 29

- Visualizzare il nome e il codice degli insegnamenti erogati nel 2010/2011 dalla facoltà di Scienze MM. FF. e NN.

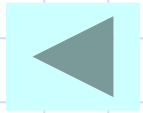
```
SELECT DISTINCT I.nomeins, I.codiceins
FROM Insegn I
INNER JOIN InsErogato IE
     ON (I.id = IE.id_insegn)
INNER JOIN Facolta F
     ON (F.id = IE.id_facolta)
WHERE F.nome = 'Scienze matematiche fisiche e naturali'
  AND IE.annoaccademico = '2010/2011'
```



Esempio 30

- Visualizzare il nome e il quadrimestre/semestre degli insegnamenti erogati nel 2006/2007 del corso di studi con id=4.
  Se l'insegnamento non ha ancora assegnato un quadrimestre/semestre deve essere comunque riportato nel risultato senza l'indicazione del quadrimestre/semestre.

```
SELECT I.nomeins, I.codiceins, PL.abbreviazione
FROM Insegn I INNER JOIN InsErogato IE
     ON I.id = IE.id_insegn
LEFT JOIN InsInPeriodo IP
     ON IE.id = IP.id_inserogato
LEFT JOIN PeriodoLez PL
     ON PL.id = IP.id_periodolez
WHERE IE.annoaccademico = '2006/2007'
  AND IE.id_corsostudi = 4
```



Uso di variabili tupla o ALIAS (1)

- SQL consente di associare un nome alternativo alle tabelle che compaiono come argomento della clausola FROM.
- Il nome viene usato per far riferimento alla tabella nel contesto dell'interrogazione e viene detto ALIAS.

```
SELECT P.Cognome
FROM Persona P
WHERE  P.Sesso = 'f';
```

- In questo caso (in cui la tabella compare una sola volta nell'interrogazione) l'ALIAS viene considerato come uno pseudonimo.

Uso di variabili tupla o ALIAS (2)

- Ogni volta che si introduce un ALIAS per una tabella si dichiara una variabile di tipo tupla che varia sul contenuto della tabella di cui è ALIAS.

```
SELECT P1.cognome
FROM Persona P1, Persona P2
WHERE P1.cognome = P2.cognome AND
      P1.id <> P2.id
```

- In questo caso (quando una tabella compare più volte nella clasusola FROM) ciascun ALIAS viene considerato come una variabile indipendente che varia sul contenuto della tabella: come se ci fossero due esemplari identici della tabella.

Esempio 31

- Trovare il cognome di tutte le persone che hanno tenuto lezioni in almeno un insegnamento dell'ateneo e per le quali esiste almeno un'altra persona nell'ateneo con lo stesso cognome.

```
SELECT P1.cognome
FROM Persona P1, Persona P2, Docente D
WHERE P1.Cognome = P2.Cognome
  AND D.id_persona = P1.id
  AND P1.id <> P2.id;
```

Interrogazioni nidificate

- SQL ammette il confronto di un valore (ottenuto come risultato di una espressione valutata sulla singola tupla) con il risultato dell'esecuzione di una interrogazione SQL.
- L'interrogazione che viene usata nel confronto viene definita direttamente nel predicato interno alla clausola WHERE.
- Si confronta un attributo con il risultato di una interrogazione.
  - PROBLEMA di disomogeneità (un insieme di valori da confrontare con un solo valore).
  - SOLUZIONE: uso delle parole chiave ALL ed ANY per estendere i normali operatori di confronto (=, >, …).

ANY ed ALL

- La parola chiave ANY specifica che la tupla corrente T soddisfa la condizione se risulta vero il confronto (con l'operatore specificato) tra il valore dell'attributo su T ed ALMENO UNO dei valori restituiti dall'interrogazione.
- La parola chiave ALL specifica che T soddisfa la condizione solo se TUTTI i valori restituiti dall'interrogazione nidificata rendono vero il confronto.
- NB: la sintassi richiede che lo schema della tabella restituita dall'interrogazione nidificata sia costituito da un solo attributo e il suo tipo sia compatibile con l'attributo con cui avviene il confronto.

Esempio 32

Visualizzare il nome degli insegnamenti che hanno un numero di crediti inferiore alla media dell'ateneo.

```
SELECT DISTINCT I.nomeins
FROM Insegn I JOIN InsErogato IE
        ON I.id = IE.id_insegn
WHERE Crediti < any (SELECT AVG(Crediti)
                     FROM InsErogato)
```

Esempio 33

- Trovare il nome degli insegnamenti (o moduli) con almeno un docente e crediti maggiori rispetto ai crediti di tutti gli insegnamenti del corso di laurea con id=6.

```
SELECT DISTINCT I.nomeins
FROM Insegn I JOIN InsErogato IE
     ON I.id = IE.id_insegn
     JOIN Docente D
     ON IE.id = D.id_inserogato
WHERE IE.crediti > all (SELECT crediti
          FROM InsErogato
          WHERE id_corsostudi = 6)
```

Interrogazioni nidificate complesse

- L'interrogazione nidificata fa riferimento al contesto dell'interrogazione che la racchiude.
- E' possibile usare nell'ambito dell'interrogazione nidificata una variabile tupla definita nell'interrogazione che la contiene.
  - Tale legame si dice "PASSAGGIO DI BINDING"
- In questo caso non vale più l'interpretazione semplice data precedentemente alle interrogazioni nidificate. Vale a dire, l'interrogazione nidificata DEVE essere valutata per ogni tupla dell'interrogazione esterna che si sta valutando.

Interrogazioni nidificate complesse – esecuzione

1. Viene costruito il prodotto cartesiano delle tabelle
2. Le condizioni che appaiono nella clausola WHERE vengono applicate a ciascuna tupla del prodotto cartesiano
3. Vengono mantenute solo le tuple per cui la condizione viene valutata vera
4. L'interrogazione nidificata (che compare all'interno di un predicato) viene valutata separatamente per ogni tupla prodotta al punto precedente.
5. Se il predicato complesso è vero la tupla viene restituita altrimenti no.

EXISTS

- L'operatore logico EXISTS ammette come parametro un'interrogazione nidificata e restituisce il valore vero solo se l'interrogazione fornisce un risultato non vuoto.
- L'operatore logico EXISTS può essere usato in modo significativo solo quando si ha un passaggio di binding tra l'interrogazione esterna e quella nidificata argomento dell'operatore.

Esempio 34

- Visualizzare il nome e il cognome dei docenti che hanno tenuto (escludento i coordinatori) nel 2010/2011 più di due insegnamenti (o moduli/unità) con più di 4 crediti.

```
SELECT P.nome, P.cognome
FROM Persona P
WHERE EXISTS (SELECT 1
        FROM Docente D JOIN InsErogato IE
                ON D.id_inserogato = IE.id
        WHERE IE.crediti > 4 AND D.coordinatore='0'
        AND IE.annoaccademico = '2010/2011'
        AND D.id_persona = P.id
        GROUP BY D.id_persona
        HAVING COUNT(*) > 2)
```

NOT EXISTS

- L’operatore logico NOT EXISTS ammette come parametro un’interrogazione nidificata e restituisce il valore vero solo se l’interrogazione fornisce un risultato vuoto.

Esempio 35

- Visualizzare il nome dei corsi di studio che nel 2006/2007 non hanno erogato insegnamenti di 4 crediti.

```
SELECT CS.nome
FROM CorsoStudi CS,
WHERE NOT EXISTS
(SELECT 1
 FROM InsErogato IE
 WHERE IE.crediti = 4
        AND IE.annoaccademico = '2006/2007'
        AND IE.id_corsostudi = CS.id)
```

Interrogazioni di tipo insiemistico

- SQL mette a disposizione anche operatori insiemistici.
- Gli operatori insiemistici si possono utilizzare solo al livello più esterno di una interrogazione SQL, vale a dire si applicano sul risultato di due o più clausole SELECT... FROM... WHERE.
- Gli operatori disponibili sono:
  - UNION
  - INTERSECT
  - EXCEPT

  e assumono come default di eseguire sempre una eliminazione dei duplicati (a meno che si usi ALL).

SelectSQL { < UNION | INTERSECT | EXCEPT >
[ALL] SelectSQL }

Esempio 36

- Visualizzare i nomi degli insegnamenti e quelli dei corsi di laurea in un unico attributo.

```
SELECT nomeins
FROM Insegn
    UNION
SELECT nome
FROM CorsoStudi;
```

Esempio 37

- Visualizzare i nomi degli insegnamenti e i nomi dei corsi di laurea che non iniziano per 'A' mantenendo i duplicati.

```
SELECT nomeins
FROM Insegn
WHERE not nomeins like 'A%'
   UNION ALL
SELECT nome
FROM CorsoStudi
WHERE not nome like 'A%';
```

Esempio 38

- Visualizzare i nomi degli insegnamenti che sono anche nomi di corsi di laurea.

```
SELECT nomeins
FROM Insegn
   INTERSECT
SELECT nome
FROM CorsoStudi;
```

Esempio 39

- Visualizzare i nomi degli insegnamenti che NON sono anche nomi di corsi di laurea.

```
SELECT nomeins
FROM Insegn
   EXCEPT
SELECT nome
FROM CorsoStudi;
```

Viste SQL

- Le viste sono tabelle “virtuali” il cui contenuto dipende dal contenuto delle altre tabelle della base di dati.
- In SQL le viste vengono definite associando un nome ed una lista di attributi ad una interrogazione SELECT...FROM...WHERE.
- Nell’interrogazione che definisce la vista possono comparire anche altre viste. SQL non ammette però:
  - dipendenze immediate (definire una vista in termini di se stessa) o ricorsive (definire una interrogazione di base e una interrogazione ricorsiva)
  - dipendenze transitive (V_1 definita usando V_2, V_2 usando V_3, …, V_n usando V_1)

Viste SQL

- Si definisce una vista usando il comando:

  CREATE VIEW NomeVista [(ListaAttributi)]

  AS SELECTSQL

- L'interrogazione SQL deve restituire un insieme di attributi pari a quello contenuto nello schema e l'ordine della target list deve corrispondere all'ordine degli attributi dello schema della vista.

- SQL permette che una vista sia aggiornabile solo quando una sola tupla di ciascuna tabella di base corrisponde a una tupla della vista.

Esempio 40

- Definire la vista che contiene le occorrenze insegnamento compresive di nomeins e codiceins presi dalla tabella Insegn.

```
CREATE VIEW InsErogatiCompleti
AS SELECT I.nomeins, I.codiceins, IE.*
FROM InsErogato IE JOIN Insegn I
 ON IE.id_insegn = I.id
```

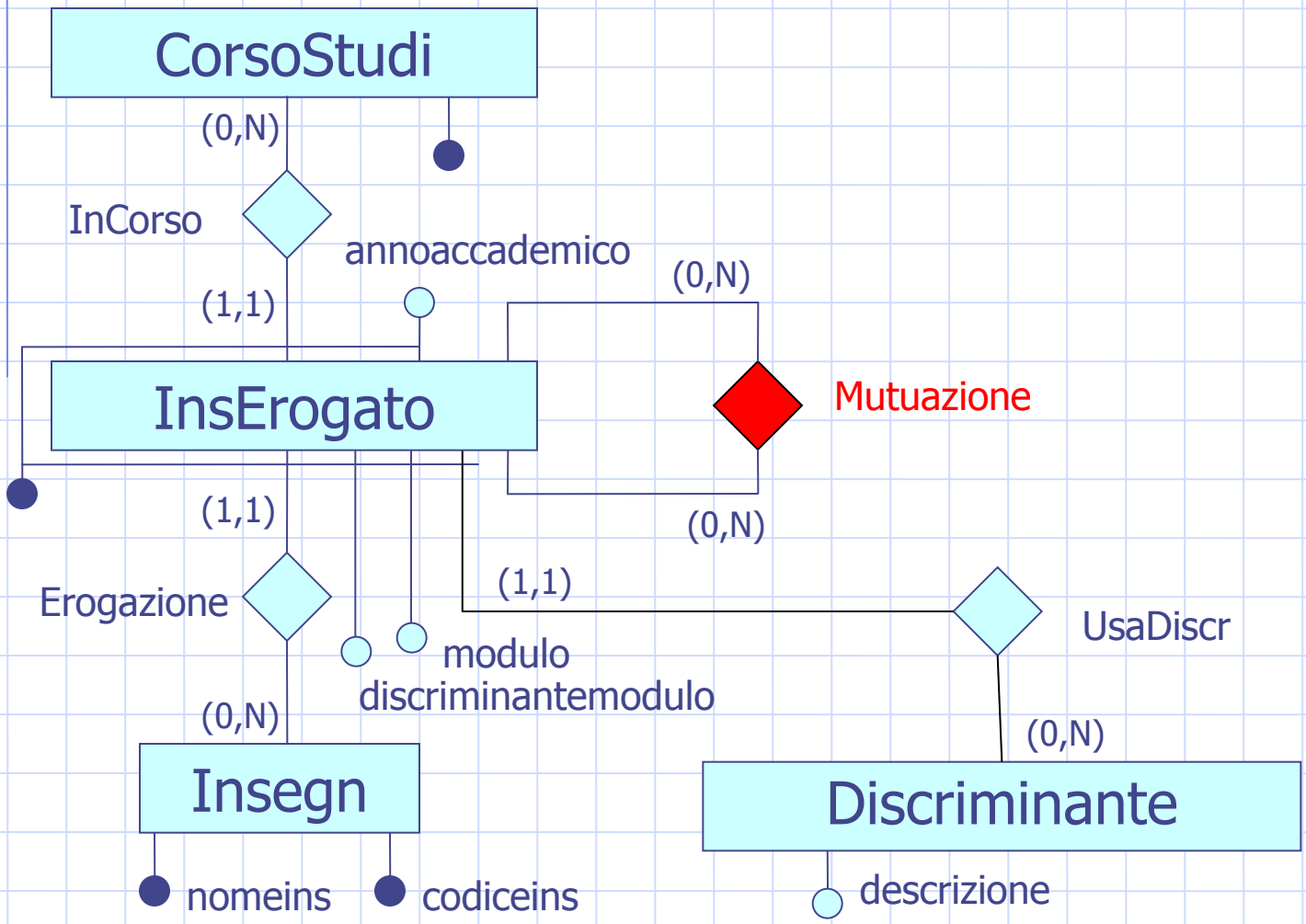
Esempio 42 (vista necessaria)

- Si vuole determinare qual è il corso di studi con il massimo numero di insegnamenti (esclusi i moduli).

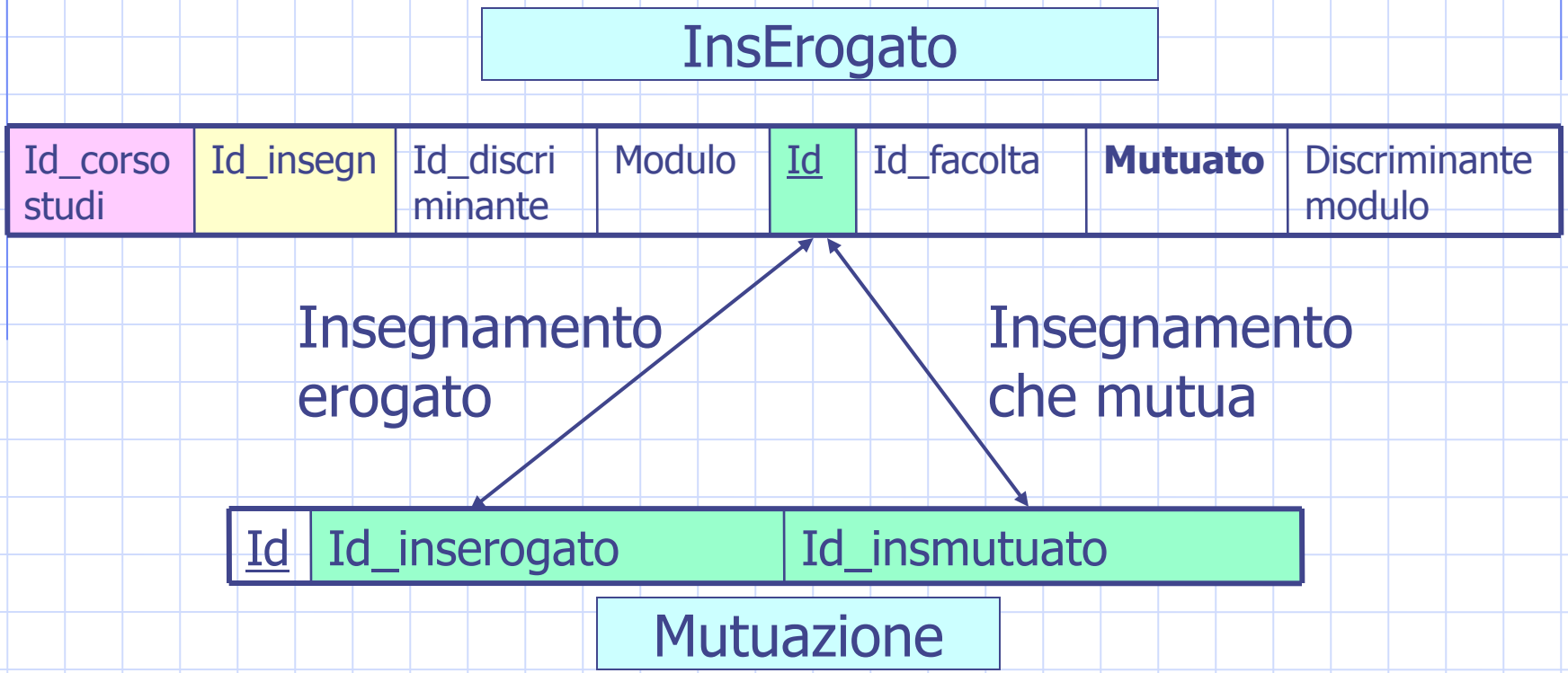
```
CREATE VIEW InsCorsoStudi(Nome, NumIns)
AS (SELECT CS.nome, count(*)
        FROM CorsoStudi CS JOIN InsErogato IE
            ON CS.id = IE.id_corsostudi
        WHERE IE.modulo = 0
        GROUP BY CS.nome)

SELECT Nome, NumIns
FROM InsCorsoStudi
WHERE NumIns = any (SELECT MAX(NumIns)
                    FROM InsCorsoStudi)
```

Schema DB ateneo - mutuazioni



Schema DB ateneo - mutuazioni



Consegna Esercitazioni 3 e 4

Inviare via email al docente due file di nome

ES3-<matricola>.sql

ES4-<matricola>.sql

Contenente tutte le interrogazioni SQL soluzioni dell'esercitazione 3 e dell'esercitazione 4. Nei file ripetere la consegna prima di ogni interrogazione SQL. Il messaggio dovrà soddisfare il seguente formato:

- Oggetto: <Matricola> - Esercitazione 3-4
- Contenuto: <Matricola> - <Cognome> - <Nome>
- Allegato: file di nome ES3-<Matricola>.sql e ES4-<Matricola>.sql

Il messaggio email va spedito entro le 23.59 del giorno 15 aprile 2013.