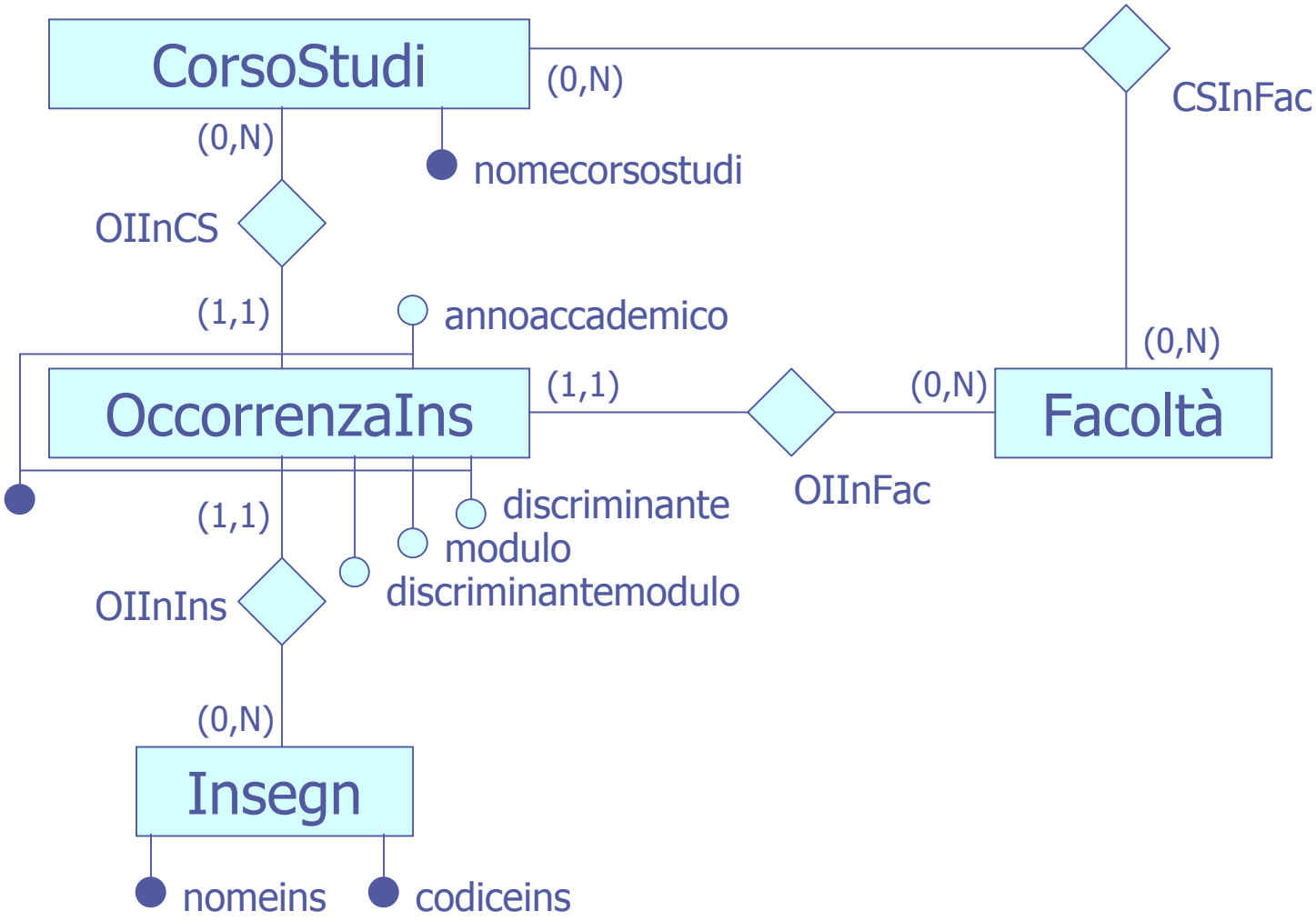
Laboratorio di Basi di Dati e Multimedia

Laurea in Tecnologie dell'Informazione: Multimedia

Docente: Alessandra Di Pierro
Email: dipierro@sci.univr.it

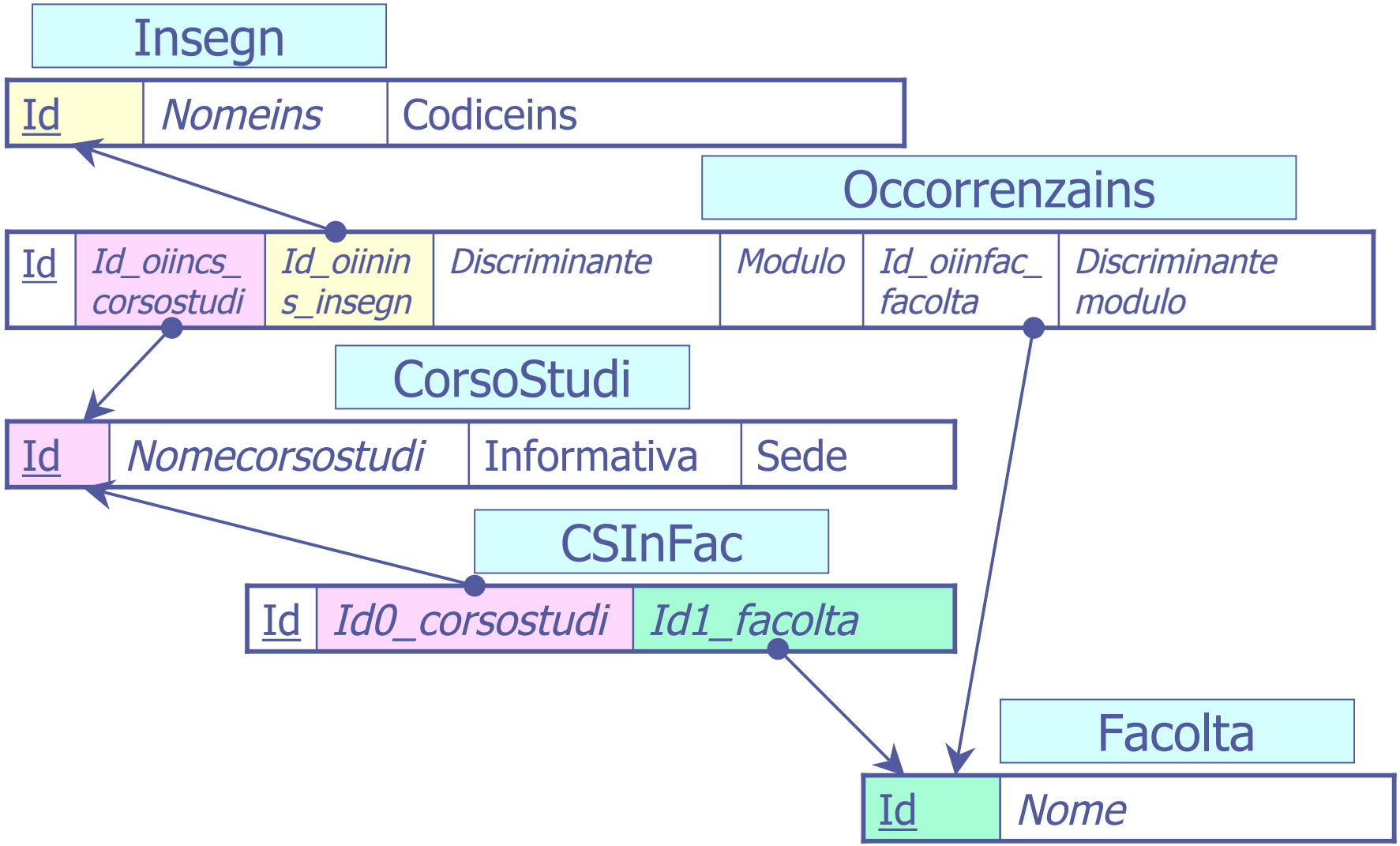
Lezione 4

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



DB dell'applicazione web di ateneo

(in corsivo gli attributi in vincolo UNIQUE)



Esempio di contenuto del DB

Occorrenzains

Id	Anno accademico	Id_oiincs_ corsostudi	Id_oiinins_ insegn	Modulo	Nome modulo	Crediti
2	2006/2007	1	34	0		10
11	2006/2007	1	34	1	Teoria	8
12	2006/2007	1	34	2	Laboratorio	2

CorsoStudi

Id	Nomecorsostudi	Sede	Informativa
1	Informatica	Verona	Xxxxxxxxxx
2	Bioinformatica	Verona	Yyyyyyyyyy

Insegn

Id	Nomeins	Codiceins
34	Basi di dati e MM	INF03
98	Programmazione	INF01

Ulteriori esempi sulle occorrenze insegnamento

Insegnamento non diviso in moduli:

- Linguaggi di programmazione (5 crediti)

Occorrenzains

Id	Anno accademico	Id_oiincs_ corsostudi	Id_oiinins _insegn	Discrimi nante	Modulo	Nome modulo	Crediti
2	2006/2007	1	8		0		5

Hamoduli	Mutuato	Id_oiinfac _facolta	Discriminante modulo		
0	0	1			

Ulteriori esempi sulle occorrenze insegnamento

Insegnamento diviso in moduli:

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

Occorrenzains

Id	Anno accademico	Id_oiincs_corsostudi	Id_oiinins_insegn	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_oiinfac_facolta	Discriminante modulo		
1	0	1			
0	0	1			
0	0	1			

Ulteriori esempi sulle occorrenze insegnamento

Insegnamento diviso in moduli replicati:

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

Occorrenzains

Id	Anno accademico	Id_oiincs_corsostudi	Id_oiinins_insegn	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_oiinfac_facolta	Discriminantemodulo
1	0	1	
0	0	1	
0	0	1	Sezione A
0	0	1	Sezione B

Ulteriori esempi sulle occorrenze insegnamento

Insegnamento replicato senza moduli:

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

Occorrenzains

Id	Anno accademico	Id_oiincs_corsostudi	Id_oiinins_insegn	Discriminante	Modulo	Nome modulo	Crediti
1	2006/2007	1	98	Sezione A	0		10
2	2006/2007	1	98	Sezione B	0		10

Hamoduli	Mutuato	Id_oiinfac_facolta	Discriminante modulo		
0	0	1			
0	0	1			

Ulteriori esempi sulle occorrenze insegnamento

Insegnamento replicato con moduli:

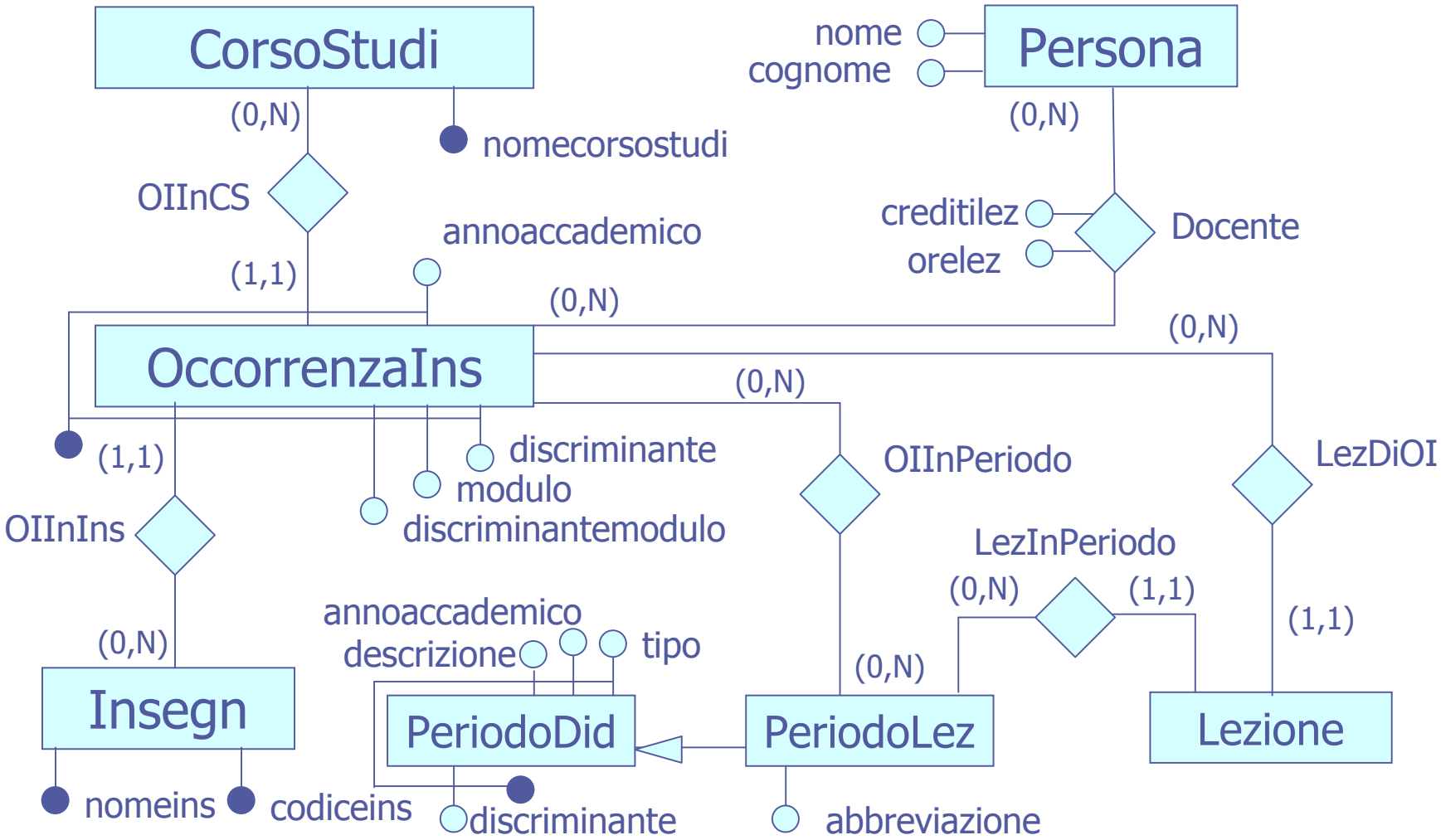
- Programmazione [Sezione A] (10 crediti)
  - Modulo Teoria (8 crediti)
  - Modulo Laboratorio [Sez A-1] (2 crediti)
  - Modulo Laboratorio [Sez A-2] (2 crediti)

Occorrenzains

Id	Anno accademico	Id_oiincs_ corsostudi	Id_oiinins _insegn	Discriminante	Modulo	Nome modulo	Crediti
1	2006/2007	1	98	Sezione A	0		10
2	2006/2007	1	98	Sezione A	0	Teoria	8
3	2006/2007	1	98	Sezione A	0	Laboratorio	2
4	2006/2007	1	98	Sezione A	0	Laboratorio	2

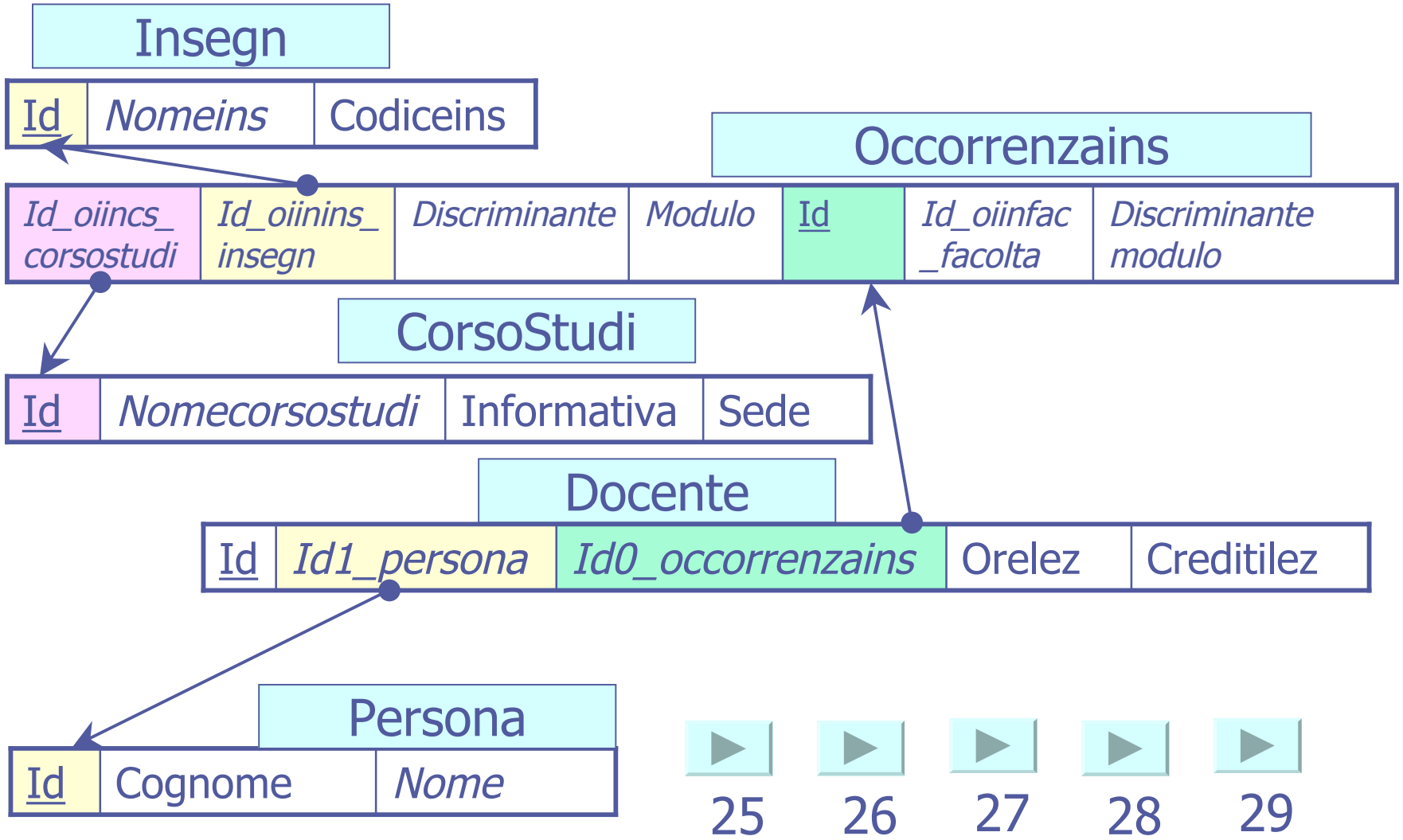
Hamoduli	Mutuato	Id_oiinfac _facolta	Discriminante modulo		
1	0	1			
0	0	1			
0	0	1	Sez A-1		
0	0	1	Sez A-2		

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



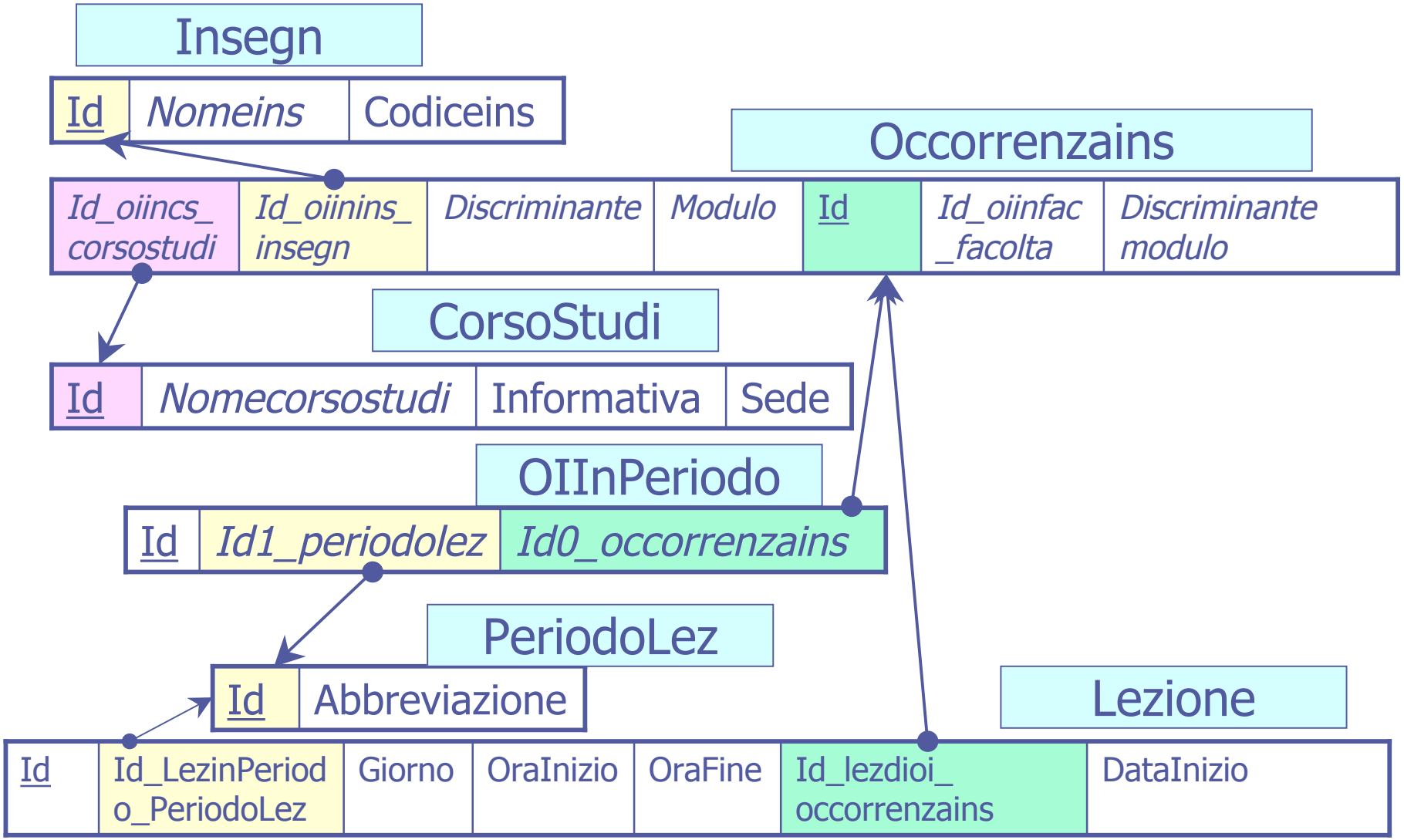
DB dell'applicazione web di ateneo

(in corsivo gli attributi in vincolo UNIQUE)



DB dell'applicazione web di ateneo

(in corsivo gli attributi in vincolo UNIQUE)



Esempio 25

- Visualizzare il nome degli insegnamenti offerti dal corso di laurea in informatica nell'anno accademico 2005/2006.

```
SELECT DISTINCT I.Nomeins
FROM   CorsoStudi CS, OccorrenzaIns OI, Insegn I
WHERE CS.id=OI.id_oiincs_corsostudi
        AND OI.id_oiinins_insegn = I.id
        AND OI.annoaccademico = '2005/2006'
        AND CS.nomecorsostudi = '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, OccorrenzaIns OI, Insegn I
WHERE P.id = D.id1_persona
        AND OI.id = D.id0_occorrenzains
        AND OI.id_oiinins_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.nomecorsostudi,
          count(DISTINCT D.id0_persona) AS NumDoc
FROM CorsoStudi CS, OccorrenzaIns OI, Docente D,
         CSInFac CSF, Facolta F
WHERE CS.id = OI.id_oiincs_corsostudi
          AND OI.id = D.id0_occorrenzains
          AND CSF.id0_corsostudi = CS.id
          AND CSF.id1_facolta = F.id
          AND OI.annoaccademico = '2006/2007'
          AND F.nome = 'Economia'
GROUP BY CS.nomecorsostudi
```

Esempio 28

- Visualizzare, per ciascun docente che tiene più di un insegnamento, il numero degli insegnamenti tenuti, e il numero totale di ore nel 2005/2006, riportando il nome e il cognome dei docenti e i conteggi richiesti.

```
SELECT P.Nome, P.Cognome
       count(D.id1_occorrenzains) AS NIns,
       sum(D.orelez) AS OreTotaliDocenza
FROM OccorrenzaIns OI, Docente D, Persona P
WHERE OI.id = D.id0_occorrenzains
       AND P.id = D.id1_persona
       AND OI.annoaccademico = '2005/2006'
GROUP BY P.id, P.Nome, P.Cognome
HAVING count(DISTINCT D.id1_occorrenzains) > 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.

inner,
right outer,
left outer,
full outer

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

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 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 2006/2007 dalla facoltà di Scienze MM. FF. e NN.

```
SELECT DISTINCT I.nomeins, I.codiceins
FROM Insegn I
        INNER JOIN OccorrenzaIns OI
                ON (I.id = OI.id_oiinins_insegn)
        INNER JOIN Facolta F
                ON (F.id = OI.id_oiinfac_facolta)
WHERE F.nome = 'Scienze matematiche fisiche e naturali'
          AND OI.annoaccademico = '2006/2007'
```

Esempio 30

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

```
SELECT I.nomeins, I.codiceins, PL.abbreviazione
FROM Insegn I INNER JOIN OccorrenzaIns OI
                ON I.id = OI.id_oiinins_insegn
        LEFT JOIN OIInPeriodo OIP
                ON OI.id = OIP.id0_occorrenzains
        JOIN PeriodoLez PL
                ON PL.id = OIP.id1_periodolez
WHERE OI.annoaccademico = '2006/2007'
        AND OI.id_oiincs_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 S.Cognome
FROM Studente S
WHERE  S.Città = 'Padova';
```

- 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 tabella in cui viene copiato il 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) l'ALIAS viene considerato come una nuova variabile.

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.id0_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 riga) 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.
- Confrontiamo 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 sia compatibile in tipo 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 I.nomeins
FROM Insegn I JOIN Occorrenzains OI
        ON I.id = OI.id_oiinins_insegn
WHERE Crediti < any (SELECT AVG(Crediti)
                     FROM Occorrenzains)
```

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 I.nomeins
FROM Insegn I JOIN Occorrenzains OI
        ON I.id = OI.id_oiinins_insegn
        JOIN Docente D
        ON OI.id = D.id0_occorrenzains
WHERE crediti > all (SELECT crediti
                     FROM Occorrenzains
                     WHERE id_oiincs_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 (2)

- Viene costruito il prodotto cartesiano delle tabelle
- Le condizione che appaiono nella clausola WHERE vengono applicate a ciascuna tupla del prodotto cartesiano
- Vengono mantenute solo le tuple per cui la condizione viene valutata vera
- L'interrogazione nidificata (che compare all'interno di un predicato) viene valutata separatamente per ogni tupla prodotta dalla valutazione dell'interrogazione più esterna.

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 nel 2005/2006 più di due insegnamenti (o moduli) con più di 4 crediti.

```
SELECT P.nome, P.cognome
FROM Persona P
WHERE EXISTS (SELECT COUNT(*)
              FROM Docente D JOIN Occorrenzains OI
                   ON D.id0_occorrenzains = OI.id
              WHERE OI.crediti > 4
                AND OI.hamoduli = 0
                AND OI.annoaccademico = '2005/2006'
                AND D.id1_persona = P.id
              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 il cui nome contiene la sottostringa 'Info'.

```
SELECT CS.nomecorsostudi
FROM CorsoStudi CS,
WHERE NOT EXISTS
        (SELECT *
         FROM Occorrenzains OI JOIN Insegn I
                ON OI.id_oiinins_insegn = I.id
         WHERE I.nomeins like '%Info%'
                AND OI.annoaccademico = '2006/2007'
                AND OI.id_oiincs_corsostudi = CS.id)
```

Interrogazioni di tipo insiemistico

- SQL mette a disposizione anche degli operatori insiemistici.
- Gli operatori insiemistici si possono utilizzare solo al livello più esterno di una query, operando sul risultato di SELECT.
- Gli operatori disponibili sono:
  - UNION
  - INTERSECT
  - EXCEPT (MINUS)

  e assumono come default di eseguire sempre una eliminazione dei duplicati (a meno dell'uso di 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 nomecorsostudi
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 nomecorsostudi
FROM CorsoStudi
WHERE not nomecorsostudi like 'A%';
```

Esempio 38

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

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

Esempio 39

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

```
SELECT nomeins
FROM Insegn
   EXCEPT
SELECT nomecorsostudi
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 al risultato dell’esecuzione di un’interrogazione.
- Nell’interrogazione che definisce la vista possono comparire anche altre viste. SQL non ammette però:
  - dipendenze ricorsive
  - dipendenze immediate (definire una vista in termini di se stessa)
  - dipendenze transitive (V1 definita usando V2, V2 usando V3, …, Vn usando V1)

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 OccorrenzeInsComplete
AS SELECT I.nomeins, I.codiceins, OI.*
FROM OccorrenzaIns OI JOIN Insegn I
    ON OI.id_oiinins_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.nomecorsostudi, count(*)
FROM CorsoStudi CS JOIN OccorrenzaIns OI
        ON CS.id = OI.id_oiincs_corsostudi
WHERE OI.modulo = 0
GROUP BY CS.nomecorsostudi)

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