

AR (2° parte) + SQL (2° parte)

BASI DI DATI I

- Algebra Relazionale
- Operazioni Insiemistiche
- Operazioni di Join (prima parte)
- Query di base in SQL
- Renaming, Distinct e Operazioni Insiemistiche

OPERAZIONI INSIEMISTICHE



OPERAZIONI INSIEMISTICHE

- Una relazione è un insieme di tuple, quindi possiamo applicare le classiche operazioni insiemistiche.
- Il risultato nella combinazione di due relazioni per mezzo di un'operazione su insieme è una nuova relazione.
- Per poter applicare un'operazione insiemistica a due relazioni, queste devono avere la stessa struttura, ovvero essere **union compatibili**...



UNION COMPATIBILITY

- Due relazioni $R(A_1, A_2, \dots, A_n)$ e $S(B_1, B_2, \dots, B_n)$ sono **union compatibili** se hanno lo stesso grado e
 $\text{dom}(A_i) = \text{dom}(B_i)$ per $1 \leq i \leq n$.



OPERAZIONI AMMISSIBILI

- Su due relazioni R ed S *union compatibili*, è possibile effettuare :
 - Unione
 - Intersezione
 - Differenza



ESEMPIO DI UNIONE

- Trovare il SSN di tutti gli impiegati che lavorano o nel dipartimento n° 5 o supervisionano direttamente un impiegato che lavora nel dipartimento n° 5:
 - $DEPS_EMPS = \sigma_{dno=5}(EMPLOYEE)$
 - $Result1 = \pi_{SSN}(DEPS_EMPS)$
 - $Result2 = \pi_{SUPERSSN}(DEPS_EMPS)$
 - $RESULT = Result1 \cup Result2$

| Result1 | 123456789 |
|---------|-----------|
| | 333444555 |
| | 666888444 |
| | 453453453 |

| Result2 | 333444555 |
|---------|-----------|
| | 888666555 |

| Result | 123456789 |
|--------|-----------|
| | 333444555 |
| | 666888444 |
| | 453453453 |
| | 888666555 |



OPERAZIONI

- **Unione:**
 - Il risultato di questa operazione, $R \cup S$, è la relazione che include tutte le tuple che sono in R o in S , oppure in R ed S . Le tuple duplicate sono eliminate.
- **Intersezione:**
 - Il risultato di questa operazione, $R \cap S$, è la relazione che include tutte le tuple che sono sia in R che in S .
- **Differenza:**
 - Il risultato di questa operazione, $R - S$, è la relazione che include tutte le tuple che sono in R ma non in S .
- Si adotta la convenzione che il risultato ha gli stessi nomi di attributi della prima relazione.



OPERAZIONI: ESEMPI

Date due relazioni S ed I

| STUDENT | FN | LN |
|---------|---------|---------|
| | Susan | Yao |
| | Ramesh | Shah |
| | Johnny | Kohler |
| | Barbara | Jones |
| | Amy | Ford |
| | Jimmy | Wang |
| | Ernest | Gilbert |

| INSTRUCTOR | FNANE | LNAME |
|------------|---------|---------|
| | John | Smith |
| | Ricardo | Browne |
| | Susan | Yao |
| | Francis | Johnson |
| | Ramesh | Shah |

| FN | LN |
|---------|---------|
| Susan | Yao |
| Ramesh | Shah |
| Johnny | Kohler |
| Barbara | Jones |
| Amy | Ford |
| Jimmy | Wang |
| Ernest | Gilbert |
| John | Smith |
| Ricardo | Browne |
| Francis | Johnson |

| FN | LN |
|--------|------|
| Susan | Yao |
| Ramesh | Shah |

$S \cap I$

| FN | LN |
|---------|---------|
| Johnny | Kohler |
| Barbara | Jones |
| Amy | Ford |
| Jimmy | Wang |
| Ernest | Gilbert |

$S - I$

| FN | LN |
|---------|---------|
| John | Smith |
| Ricardo | Browne |
| Francis | Johnson |

$I - S$

$S \cup I$



PROPRIETÀ DELLE OPERAZIONI

- Unione ed intersezione sono commutative, associative e possono essere applicate ad un numero qualsiasi di relazioni
 - $R \cup S = S \cup R$
 - $R \cap S = S \cap R$
 - $R \cup (S \cup T) = (R \cup S) \cup T$
 - $R \cap (S \cap T) = (R \cap S) \cap T$
- La differenza non è commutativa.
 - In generale $R - S \neq S - R$.



PRODOTTO CARTESIANO (CROSS PRODUCT O CROSS JOIN)

- Non è necessario che le relazioni siano union compatibili
 - $R(A_1, A_2, \dots, A_n) \times S(B_1, B_2, \dots, B_m) = Q(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m)$
 - In Q si ha una tupla per ogni combinazione di una da R ed una da S.
 - Se R contiene n_r tuple ed S contiene n_s tuple, allora $R \times S$ contiene $n_r \cdot n_s$ tuple.
- In caso di attributi con lo stesso nome nelle due relazioni, si deve effettuare il rename di uno dei due.



PRODOTTO CARTESIANO: ESEMPIO

- Vogliamo trovare per ogni impiegato di sesso femminile, una lista dei suoi familiari a carico:
 - $\text{FEMALE_EMPS} = \sigma_{\text{Sex}=\text{"F"}}(\text{EMPLOYEE})$
 - $\text{MPNAMES} = \pi_{<\text{FNAME}, \text{LNAME}, \text{SSN}>}(\text{FEMALE_EMPS})$
 - $\text{EMP_DEPENDENTS} = \text{MPNAMES} \times \text{DEPENDENTS}$
 - $\text{ACTUAL_DEPENDENTS} = \sigma_{\text{SSN}=\text{ESSN}}(\text{EMP_DEPENDENTS})$
 - $\text{RESULT} = \pi_{<\text{FNAME}, \text{LNAME}, \text{DEPENDENT_NAME}>}(\text{ACTUAL_DEPENDENTS})$



PRODOTTO CARTESIANO: ESEMPIO (2)

| FEMALE_EMPS | FNAME | MINIT | LNAME | SSN | BDATE | ADDRESS | SEX | SALARY | SUPERSSN | DNO |
|-------------|----------|-------|---------|-----------|------------|-------------------------|-----|--------|-----------|-----|
| | Alicia | J | Zelaya | 999887777 | 1968-07-19 | 3321 Castle, Spring, TX | F | 25000 | 987654321 | 4 |
| | Jennifer | S | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire, TX | F | 43000 | 888865555 | 4 |
| | Joyce | A | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX | F | 25000 | 333445555 | 5 |

| EMPNAMEs | FNAME | LNAME | SSN |
|----------|----------|---------|-----------|
| | Alicia | Zelaya | 999887777 |
| | Jennifer | Wallace | 987654321 |
| | Joyce | English | 453453453 |

| EMP_DEPENDENTS | FNAME | LNAME | SSN | ESSN | DEPENDENT_NAME | SEX | BDATE | *** |
|----------------|----------|---------|-----------|-----------|----------------|-----|------------|-----|
| | Alicia | Zelaya | 999887777 | 333445555 | Alice | F | 1986-04-05 | *** |
| | Alicia | Zelaya | 999887777 | 333445555 | Theodore | M | 1983-10-25 | *** |
| | Alicia | Zelaya | 999887777 | 333445555 | Joy | F | 1958-05-03 | *** |
| | Alicia | Zelaya | 999887777 | 987654321 | Abner | M | 1942-02-28 | *** |
| | Alicia | Zelaya | 999887777 | 123456789 | Michael | M | 1988-01-04 | *** |
| | Alicia | Zelaya | 999887777 | 123456789 | Alice | F | 1988-12-30 | *** |
| | Alicia | Zelaya | 999887777 | 123456789 | Elizabeth | F | 1967-05-05 | *** |
| | Jennifer | Wallace | 987654321 | 333445555 | Alice | F | 1986-04-05 | *** |
| | Jennifer | Wallace | 987654321 | 333445555 | Theodore | M | 1983-10-25 | *** |
| | Jennifer | Wallace | 987654321 | 333445555 | Joy | F | 1958-05-03 | *** |
| | Jennifer | Wallace | 987654321 | 987654321 | Abner | M | 1942-02-28 | *** |
| | Jennifer | Wallace | 987654321 | 123456789 | Michael | M | 1988-01-04 | *** |
| | Jennifer | Wallace | 987654321 | 123456789 | Alice | F | 1988-12-30 | *** |
| | Jennifer | Wallace | 987654321 | 123456789 | Elizabeth | F | 1967-05-05 | *** |
| | Joyce | English | 453453453 | 333445555 | Alice | F | 1986-04-05 | *** |
| | Joyce | English | 453453453 | 333445555 | Theodore | M | 1983-10-25 | *** |
| | Joyce | English | 453453453 | 333445555 | Joy | F | 1958-05-03 | *** |
| | Joyce | English | 453453453 | 987654321 | Abner | M | 1942-02-28 | *** |
| | Joyce | English | 453453453 | 123456789 | Michael | M | 1988-01-04 | *** |
| | Joyce | English | 453453453 | 123456789 | Alice | F | 1988-12-30 | *** |
| | Joyce | English | 453453453 | 123456789 | Elizabeth | F | 1967-05-05 | *** |

| ACTUAL_DEPENDENTS | FNAME | LNAME | SSN | ESSN | DEPENDENT_NAME | SEX | BDATE |
|-------------------|----------|---------|-----------|-----------|----------------|-----|------------|
| | Jennifer | Wallace | 987654321 | 987654321 | Abner | M | 1942-02-28 |

| RESULT | FNAME | LNAME | DEPENDENT_NAME |
|--------|----------|---------|----------------|
| | Jennifer | Wallace | Abner |



PRODOTTO CARTESIANO

- Operazione binaria
- Contiene sempre un numero di n -ple pari al prodotto delle cardinalità degli operandi (le n -ple sono tutte combinabili).



Impiegati

| Impiegato | Reparto |
|-----------|---------|
| Rossi | A |
| Neri | B |
| Bianchi | B |

Reparti

| Codice | Capo |
|--------|-------|
| A | Mori |
| B | Bruni |

Impiegati × Reparti

| Impiegato | Reparto | Codice | Capo |
|-----------|---------|--------|-------|
| Rossi | A | A | Mori |
| Rossi | A | B | Bruni |
| Neri | B | A | Mori |
| Neri | B | B | Bruni |
| Bianchi | B | A | Mori |
| Bianchi | B | B | Bruni |



OPERAZIONI DI JOIN (1° PARTE)



JOIN

- La sequenza di operazioni appena vista, riassumibile in $\sigma_{join-condition}(R \times S)$ è abbastanza frequente: per questo è stata creata un'operazione speciale, chiamata JOIN.
- La JOIN, denotata con \bowtie , è usata per combinare tuple relate in una sola tupla.
 - $\sigma_{join-condition}(R \times S)$ diventa $R \bowtie_{join-condition} S$



JOIN: ESEMPIO

- Supponiamo di voler trovare il nome del manager di ciascun dipartimento:
 - Combiniamo ogni tupla dipartimento con la tupla impiegato il cui SSN fa match col valore MGRSSN nella tupla dipartimento.
- DEPT_MGR= department $\bowtie_{MGRSSN=SSN}$ EMPLOYEE
- RESULT= $\pi_{DNAME, LNAME, FNAME}(\text{DEPT_MGR})$

| DEPT_MGR | DNAME | DNUMBER | MGRSSN | • • • | FNAME | MINIT | LNAME | SSN | • • • |
|----------------|-------|-----------|--------|----------|-------|---------|-----------|-------|-------|
| Research | 5 | 333445555 | • • • | Franklin | T | Wong | 333445555 | • • • | • • • |
| Administration | 4 | 987654321 | • • • | Jennifer | S | Wallace | 987654321 | • • • | • • • |
| Headquarters | 1 | 888665555 | • • • | James | E | Borg | 888665555 | • • • | • • • |



JOIN: ESEMPIO SU PRODOTTO CARTESIANO

- L'esempio precedente sul prodotto cartesiano può essere risolto rimpiazzando le operazioni:

- $\text{EMP_DEPENDENTS} = \text{EMPNAME} \times \text{DEPENDENTS}$
- $\text{ACTUAL_DEPENDENTS} = \sigma_{\text{SSN}=\text{ESSN}}(\text{EMP_DEPENDENTS})$

Con:

- $\text{ACTUAL_DEPENDENTS} = \text{EMPNAME} \bowtie_{\text{SSN}=\text{ESSN}} \text{DEPENDENTS}$



QUERY DI BASE IN SQL



SQL E GLI INSIEMI

- Occorre fare un'importante distinzione tra SQL ed il modello relazionale formale:
 - SQL consente di avere **più tuple identiche** in tutti gli attributi, quindi in generale una tabella SQL **non è un insieme** di tuple.
È invece un **multiset** (o bag) di tuple.
 - Alcune relazioni possono essere vincolate ad essere insiemi, usando il vincolo di chiave oppure l'opzione **DISTINCT** con lo statement SELECT...



SQL, OPERAZIONI SUI DATI

- interrogazione:
 - **SELECT**
- modifica:
 - **INSERT, DELETE, UPDATE**



IL COMANDO SELECT

- Il comando **SELECT** è l'istruzione di base per recuperare informazioni da un database.
- Il SELECT dell'SQL non ha relazioni con l'operatore di select dell'algebra relazionale.
- La forma di base, detta mapping o blocco di **SELECT FROM WHERE** è formata da tre clausole:

```
SELECT <lista_attributi>
FROM <lista_tabelle>
WHERE <condizione>
```

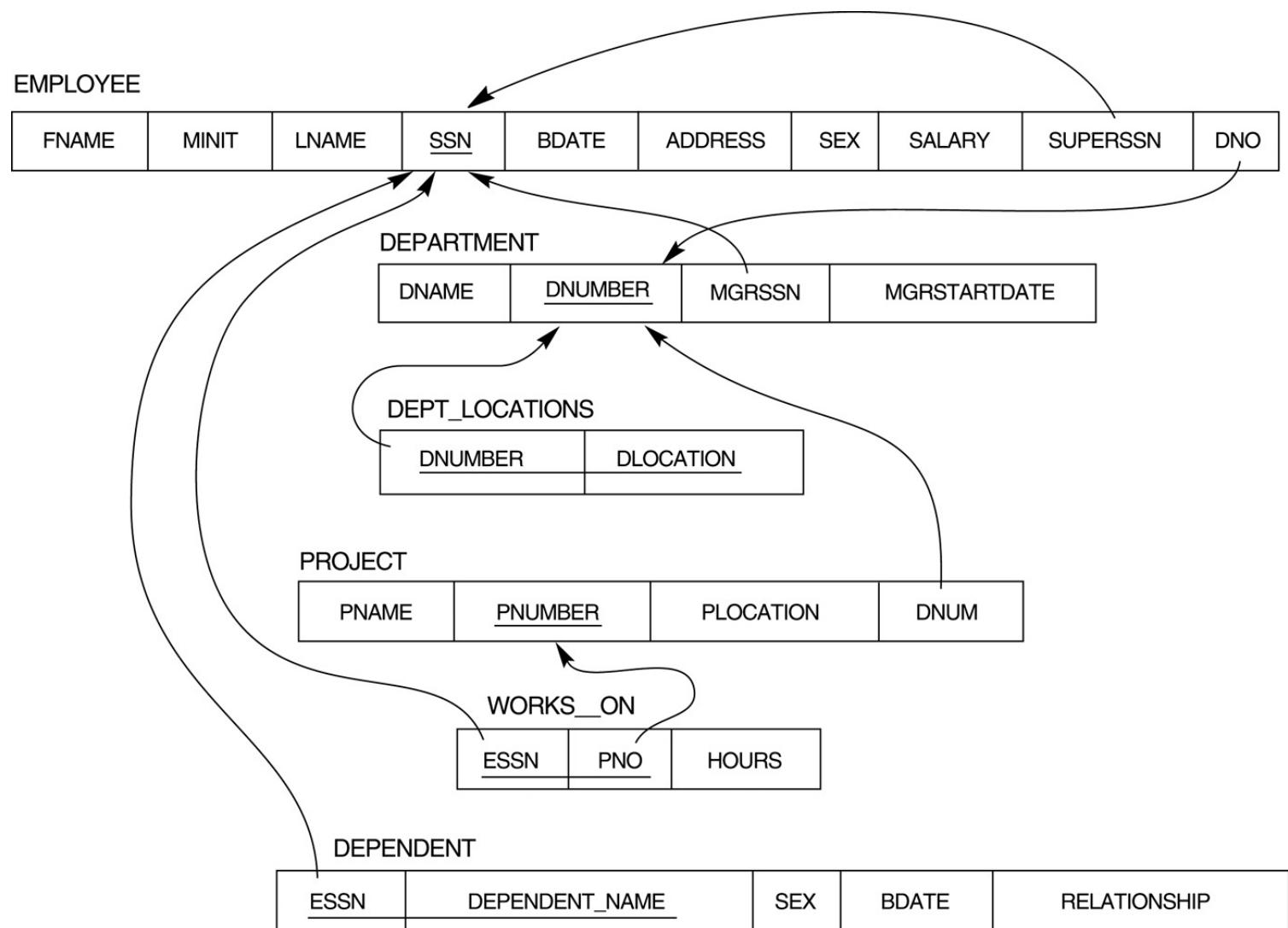


IL COMANDO SELECT (2)

- **<lista_attributi>** è una lista di nomi di attributi i cui valori devono essere recuperati dalla query.
- **<lista_tabelle>** è una lista di nomi di relazioni richiesti per elaborare la query.
- **<condizione>** è un'espressione booleana di ricerca che identifica la tupla da ritrovare.



MAPPING DELLO SCHEMA COMPANY



IL COMANDO SELECT: ESEMPIO

- Trovare la data di nascita e l'indirizzo dell'impiegato di nome '*John B. Smith*':

```
SELECT      BDATE, ADDRESS  
FROM        EMPLOYEE  
WHERE       FNAME='JOHN' AND MINIT='B' AND  
           LNAME='SMITH';
```

- BDATE e ADDRESS sono detti anche **Attributi di proiezione**.
- Equivalente nell'algebra relazionale:

$$\pi_{\langle \text{BDATE}, \text{ADDRESS} \rangle} (\sigma_{\text{FNAME}='JOHN' \text{ AND } \text{MINIT}='B'} (\text{EMPLOYEE})) \\ \text{AND } \text{LNAME}='SMITH'$$


IL COMANDO SELECT: ESEMPIO (2)

- Trovare cognome, nome e indirizzo di tutti gli impiegati del dipartimento '*Research*':

```
SELECT    FNAME, LNAME, ADDRESS  
FROM      EMPLOYEE, DEPARTMENT  
WHERE    DNAME='Research' AND DNUMBER=DNO;
```

- È simile alla sequenza SELECT-PROJECT-JOIN dell'algebra relazionale, ed è perciò detta query *select-project-join*.
- Equivalente nell'algebra relazionale:

$$\pi_{\langle \text{FNAME}, \text{LNAME}, \text{ADDRESS} \rangle} (\sigma_{\text{DNAME}=\text{'Research'}} (\text{EMPLOYEE} \bowtie_{\text{DNO}=\text{DNUMBER}} \text{DEPARTMENT}))$$


Maternità

| Madre | Figlio |
|-------|---------|
| Luisa | Maria |
| Luisa | Luigi |
| Anna | Olga |
| Anna | Filippo |
| Maria | Andrea |
| Maria | Aldo |

Paternità

| Padre | Figlio |
|--------|---------|
| Sergio | Franco |
| Luigi | Olga |
| Luigi | Filippo |
| Franco | Andrea |
| Franco | Aldo |

Persone

| Nome | Età | Reddito |
|---------|-----|---------|
| Andrea | 27 | 21 |
| Aldo | 25 | 15 |
| Maria | 55 | 42 |
| Anna | 50 | 35 |
| Filippo | 26 | 30 |
| Luigi | 50 | 40 |
| Franco | 60 | 20 |
| Olga | 30 | 41 |
| Sergio | 85 | 35 |
| Luisa | 75 | 87 |



SELEZIONE E PROIEZIONE

- Nome e reddito delle persone con gli anni minore o uguale a trenta:

$$\pi_{\langle \text{nome, reddito} \rangle}(\sigma_{\text{eta} \leq 30}(\text{Persone}))$$

```
SELECT nome, reddito  
FROM persone  
WHERE eta <= 30
```

- Abbreviazione di:

```
SELECT p.nome AS nome, p.reddito AS reddito  
FROM persone p  
WHERE p.eta <= 30
```



(animazione)

Persone

| Nome | Reddito |
|---------|---------|
| Andrea | 21 |
| Aldo | 15 |
| Filippo | 30 |



SELEZIONE, PROIEZIONE E JOIN

- Istruzioni SELECT con una sola relazione nella clausola FROM permettono di realizzare:
 - selezioni, proiezioni, ridenominazioni.
- Con più relazioni nella FROM si realizzano join (e prodotti cartesiani).



SQL E ALGEBRA RELAZIONALE

- $R1(A1, A2) \ R2(A3, A4)$

```
SELECT R1.A1, R2.A4  
FROM R1, R2  
WHERE R1.A2 = R2.A3
```

- proiezione (**SELECT**)
- prodotto cartesiano (**FROM**)
- selezione (**WHERE**)



SQL E ALGEBRA RELAZIONALE (2)

- Consideriamo gli schemi $R1(A1, A2)$ $R2(A3, A4)$:

```
SELECT R1.A1, R2.A4  
FROM      R1, R2  
WHERE    R1.A2 = R2.A3
```

$$\pi_{\langle A1, A4 \rangle} (\sigma_{A2=A3} (R_1 \times R_2))$$


PROIEZIONE SENZA SELEZIONE

- Nome e reddito di tutte le persone :

$\pi_{<\text{nome, reddito}>}(\text{Persone})$

```
SELECT nome, reddito  
FROM persone
```

- Abbreviazione di:

```
SELECT p.nome AS nome, p.reddito AS reddito  
FROM persone p
```



ABBREVIAZIONI

- Dato uno schema R(A,B); tutti gli attribuiti di R:

$$\pi_{A, B}(R)$$

```
SELECT *
FROM R
```

- equivale (intuitivamente) a:

```
SELECT X.A AS A, X.B AS B
FROM R X
WHERE TRUE
```



ESEMPIO

- I padri di persone che guadagnano più di 22:

$$\pi_{\text{Padre}} (\text{Paternita} \bowtie_{\text{Figlio}=\text{Nome}} \sigma_{\text{Reddito} > 22} (\text{Persone}))$$

```
SELECT      DISTINCT Padre  
FROM        Persone, Paternita  
WHERE       Figlio = Nome AND Reddito > 22
```

| Padre |
|-------|
| Luigi |
| Luigi |

| Padre |
|-------|
| Luigi |



IL COMANDO SELECT: ESEMPIO

- Per ogni progetto localizzato a '*Strafford*', listare il n° di progetto, il n° di dipartimento di controllo, ed il cognome, l'indirizzo e la data di nascita del manager del dipartimento:

```
SELECT      PNUMBER, DNUM, LNAME, ADDRESS, BDATE  
FROM        PROJECT, DEPARTMENT, EMPLOYEE  
WHERE       DNUM=DNUMBER AND MGRSSN=SSN AND PLOCATION='Stafford'
```



RENAMING, DISTINCT E OPERAZIONI INSIEMISTICHE



NOMI DI ATTRIBUTI E RENAMING

- In SQL lo stesso nome può essere usato per più attributi solo se questi appartengono a relazioni diverse.
- Se una query coinvolge tali relazioni, occorre **qualificare** il nome dell'attributo con il nome della relazione per evitare ambiguità.
- **Esempio:** Employee.SSN



NOMI DI ATTRIBUTI E RENAMING (2)

- Si può avere ambiguità anche nel caso di query che riferiscono due volte alla stessa relazione:
 - **Esempio:** Per ogni impiegato, trovare il suo nome il suo cognome e quello del suo diretto superiore:

```
SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE AS E, EMPLOYEE AS S
WHERE E.SUPERSSN=S.SSN;
```
- Abbiamo dichiarato nomi di relazione alternativi **E** ed **S**, detti **alias**, per la relazione EMPLOYEE .



NOMI DI ATTRIBUTI E RENAMING (3)

- È anche possibile rinominare gli attributi della relazione nella query, dando loro degli alias scrivendo:
EMPLOYEE AS E(FN, MI, LN, SSN, BD, ADDR, SEX, SAL, SSSN, DNO)
- nella clausola **FROM**.
- *Quella che abbiamo visto è un esempio di query ricorsiva.*



MANCANZA DEL WHERE

- Omettere la clausola WHERE equivale a **WHERE TRUE**, cioè tutte le tuple della relazione specificata nella clausola FROM fanno parte del risultato.
- Se più di una relazione è specificata nella clausola **FROM**, allora il risultato sarà il *prodotto cartesiano* delle relazioni.



MANCANZA DEL WHERE: ESEMPI

- *Esempio:* Selezionare tutti i SSN:

```
SELECT SSN  
FROM EMPLOYEE;
```

- *Esempio:* Selezionare tutte le combinazioni Employee.SSN e Department.Dname:

```
SELECT SSN, DNAME  
FROM EMPLOYEE, DEPARTMENT;
```



IL CARATTERE JOLLY “*”

- Per recuperare tutti gli attributi delle tuple selezionate, si usa il carattere jolly *:

- **Esempio:** Trovare tutti i valori degli attributi degli impiegati che lavorano per il dipartimento n°5:

```
SELECT *  
FROM EMPLOYEE  
WHERE DNO=5;
```

- **Esempio:** Trovare tutti gli attributi di Employee e gli attributi di Department per cui lavora ogni impiegato del dipartimento ‘*Research*’:

```
SELECT *  
FROM EMPLOYEE, DEPARTMENT  
WHERE DNAME='Research' AND DNO=DNUMBER;
```



DUPLICAZIONI DI TUPLE IN SQL

- SQL non tratta relazioni come insiemi: *tuple duplicate possono apparire più di una volta.*
- SQL non elimina le duplicazioni per le seguenti ragioni:
 - è un'operazione costosa (l'implementazione richiederebbe l'ordinamento e poi l'eliminazione);
 - l'utente può essere interessato alle duplicazioni;
 - con funzioni di aggregazione siamo interessati a non eliminarle.



LA CLAUSOLA DISTINCT

- Se le duplicazioni non sono volute, lo si specifica con la clausola **DISTINCT**:

- **Esempi:**

- Trovare i salari di tutti gli impiegati:

```
SELECT SALARY  
FROM EMPLOYEE;
```

- Trovare i salari distinti degli impiegati:

```
SELECT DISTINCT SALARY  
FROM EMPLOYEE;
```



DISTINCT, ATTENZIONE

- cognome e filiale di tutti gli impiegati

| Cognome | Filiale |
|---------|---------|
| Neri | Napoli |
| Neri | Milano |
| Rossi | Roma |

$\pi_{<\text{Cognome}, \text{Filiale}>} (\text{Impiegati})$



SELECT
Cognome, Filiale
FROM Impiegati

| Cognome | Filiale |
|---------|---------|
| Neri | Napoli |
| Neri | Milano |
| Rossi | Roma |
| Rossi | Roma |

SELECT DISTINCT
Cognome, Filiale
FROM Impiegati

| Cognome | Filiale |
|---------|---------|
| Neri | Napoli |
| Neri | Milano |
| Rossi | Roma |



OPERAZIONI INSIEMISTICHE

- SQL incorpora le seguenti operazioni insiemistiche (è *richiesta la union-compatibilità*):
 - UNION
 - EXCEPT
 - INTERSECT
- **EXCEPT** restituisce tutti i valori distinti della query a sinistra dell'operando non presenti nella query a destra.
- Usando tali operazioni le tuple duplicate sono eliminate (a meno che non venga richiesto il contrario con la clausola **ALL**).

} SQL 2



OPERAZIONI INSIEMISTICHE - ESEMPIO

- Fare una lista dei numeri di progetti per i progetti che coinvolgono un impiegato il cui cognome è ‘Smith’ come lavoratore **oppure** come manager del dipartimento che controlla il progetto:

```
(SELECT PNUMBER
      FROM PROJECT,WORKS_ON,EMPLOYEE
     WHERE PNUMBER=PNO AND ESSN=SSN AND
          LNAME='Smith');

UNION

(SELECT PNUMBER
      FROM PROJECT,DEPARTMENT,EMPLOYEE
     WHERE DNUM=DNUMBER AND MGRSSN=SSN AND
          LNAME='Smith')
```



CONFRONTO TRA SOTTOSTRINCHE

- Per il confronto tra stringhe si usa l'operatore **LIKE**
- *Caratteri jolly:*
 - '%' rimpiazza qualsiasi numero di caratteri;
 - '_' rimpiazza un singolo carattere;
- **Esempio:** Trovare tutti gli impiegati il cui indirizzo è a '*Houston, Texas*':

```
SELECT FNAME, LNAME  
FROM EMPLOYEE  
WHERE ADDRESS LIKE '%HOUSTON, TEXAS%';
```



“LIKE” - ESEMPIO

- Le persone che hanno un nome che inizia per '*A*' e ha una '*d*' come terza lettera:

```
SELECT *
FROM persone
WHERE nome LIKE 'A_d%';
```



CONFRONTO TRA SOTTOSTRINGHE - *ESEMPI*

- Trovare tutti gli impiegati nati negli anni '50. Il formato di data è *YYYY-MM-DD*:

```
SELECT FNAME, LNAME  
FROM EMPLOYEE  
WHERE BDATE LIKE '__ 5 _____';
```

- Mostrare i salari risultanti se a tutti gli impiegati che lavorano sul progetto '*Product X*' viene concesso un aumento del 10%:

```
SELECT FNAME, LNAME, 1.1*SALARY  
FROM EMPLOYEE, WORKS_ON, PROJECT  
WHERE ESSN=SSN AND PNO=PNUMBER AND PNAME='Product X';
```



GESTIONE DEI VALORI NULLI

Impiegati

| Matricola | Cognome | Filiale | Età |
|-----------|---------|---------|------|
| 5998 | Neri | Milano | 45 |
| 9553 | Bruni | Milano | NULL |

- Gli impiegati la cui età è NULL o potrebbe essere maggiore di 40.

σ Età > 40 OR Età IS NULL (Impiegati)



GESTIONE DEI VALORI NULLI (2)

- Gli impiegati la cui età è NULL o potrebbe essere maggiore di 40.

σ Età > 40 OR Età IS NULL (Impiegati)

```
SELECT *
FROM Impiegati
WHERE eta > 40 OR eta IS NULL
```





FINE

Per eventuali domande: (in ordine di preferenza personale)

- Ora.
- Chat di Teams
- Mail: silvio.barra@unina.it

