

# SQL

## **Laboratorio di** ***Progettazione di Basi di Dati*** (CdS in Informatica e TPS)

**a.a. 2013/2014**

<http://www.di.uniba.it/~lisi/courses/basi-dati/bd2013-14.htm>

dott.ssa Francesca A. Lisi  
lisi@di.uniba.it

Orario di ricevimento: giovedì ore 10-12

# Sommario (V parte)

- Aspetti avanzati della manipolazione dei dati
- Aspetti avanzati della definizione dei dati
- Controllo di accesso
- Gestione delle transazioni

## Riferimenti

- capp. 5-6 di Pratt
- capp. 9-13 di “MySQL Tutorial”
- cap. 5, in particolare 5.1-5.4, di Atzeni et al.
- cap. 9, in particolare 9.1-9.2, di Elmasri & Navathe

# Interrogazioni e manipolazione dati

```
INSERT INTO Persone (Nome)
  SELECT Padre
  FROM Paternita
  WHERE Padre NOT IN
    (SELECT Nome FROM Persone)
```

```
DELETE FROM Paternita
WHERE Figlio NOT IN
  (SELECT Nome FROM Persone)
```

# Vincoli di integrità generici

La clausola **CHECK** specifica vincoli di integrità a livello di ennupla (e anche vincoli più complessi)

```
CREATE TABLE Impiegato(  
  Matricola CHARACTER(6),  
  Cognome CHARACTER(20),  
  Nome CHARACTER(20),  
  Sesso CHARACTER NOT NULL  
    CHECK (Sesso IN ('M', 'F')),  
  Stipendio INTEGER,  
  Superiore CHARACTER(6),  
  CHECK (Stipendio <= (  
    SELECT Stipendio  
    FROM Impiegato J  
    WHERE Superiore = J.Matricola))  
)
```

# Asserzioni

Le asserzioni specificano vincoli a livello di schema

```
CREATE ASSERTION AlmenoUnImpiegato  
  CHECK (1 <= (  
    SELECT COUNT(*)  
    FROM Impiegato)  
  )
```

# Viste

```
CREATE VIEW NomeVista  
[ (ListaAttributi) ] AS SelectSQL  
[WITH [ LOCAL | CASCADE ] CHECK OPTION]
```

```
CREATE VIEW ImpiegatiAmmin  
  (Matricola, Nome, Cognome, Stipendio) AS  
  SELECT Matricola, Nome, Cognome, Stipendio  
  FROM Impiegato  
  WHERE Dipart = 'Amministrazione' AND  
         Stipendio > 10
```

## Viste (II)

- Gli aggiornamenti sono ammessi (di solito) solo su viste definite su una sola relazione
- Alcune verifiche possono essere imposte

```
CREATE VIEW ImpiegatiAmminPoveri AS
  SELECT *
  FROM ImpiegatiAmmin
  WHERE Stipendio < 50
  WITH CHECK OPTION
```

**CHECK OPTION** permette modifiche, ma solo a condizione che la ennupla continui ad appartenere alla vista (non posso modificare lo stipendio portandolo a 60)

# Viste per la scrittura di interrogazioni

Es. Estrarre il dipartimento caratterizzato dal massimo della somma degli stipendi

```
SELECT Dipart  
FROM Impiegato  
GROUP BY Dipart  
HAVING SUM(Stipendio) >= ALL (  
    SELECT SUM(Stipendio)  
    FROM Impiegato  
    GROUP BY Dipart  
)
```



# Viste per la scrittura di interrogazioni

## CREATE VIEW

```
BudgetStipendi (Dip, TotaleStipendi) AS  
SELECT Dipart, SUM(Stipendio)  
FROM Impiegato  
GROUP BY Dipart
```

```
SELECT Dip  
FROM BudgetStipendi  
WHERE TotaleStipendi =(  
    SELECT MAX(TotaleStipendi)  
    FROM BudgetStipendi)
```

# Viste per la scrittura di interrogazioni

Es. Estrarre il numero medio di uffici per dipartimento

```
SELECT AVG (COUNT (DISTINCT Ufficio))  
FROM Impiegato  
GROUP BY Dipart
```

## CREATE VIEW

```
DipartUffici (NomeDip, NroUffici) AS  
SELECT Dipart, COUNT (DISTINCT Ufficio)  
FROM Impiegato  
GROUP BY Dipart;
```

```
SELECT AVG (NroUffici)  
FROM DipartUffici
```

# Funzioni scalari

- Funzioni a livello di ennupla che restituiscono singoli valori
- Temporali
  - `current_date`, `extract (year from ...)`
- Manipolazione stringhe
  - `char_length`, `lower`
- Conversione
  - `cast`
- Condizionali
  - `case`, `coalesce`, `nullif`

# Funzioni condizionali

```
SELECT Nome, Cognome,  
       coalesce(Dipart, 'Ignoto')  
FROM Impiegato
```

```
SELECT Targa,  
       CASE Tipo  
         WHEN 'Auto' THEN 2.58 * KWatt  
         WHEN 'Moto' THEN (22.00 + 1.00 * KWatt)  
         ELSE NULL  
       END AS Tassa  
FROM Veicolo  
WHERE Anno > 1975
```

# Controllo dell'accesso

- In SQL è possibile specificare chi (utente) e come (lettura, scrittura, ...) può utilizzare la base di dati (o parte di essa)
- Oggetto dei *privilegi* (diritti di accesso) sono di solito le tabelle, ma anche altri tipi di *risorse*, quali singoli attributi, viste o domini
- Un utente predefinito `_system` (amministratore della base di dati) ha tutti i privilegi
- Il creatore di una risorsa ha tutti i privilegi su di essa

# Privilegi

- Un privilegio è caratterizzato da:
  - la risorsa cui si riferisce
  - l'utente che concede il privilegio
  - l'utente che riceve il privilegio
  - l'azione che viene permessa
  - la trasmissibilità del privilegio

# Tipi di privilegi

- **INSERT**: permette di inserire nuove ennuple in tabelle e viste
- **UPDATE**: permette di modificare ennuple
- **DELETE**: permette di eliminare ennuple
- **SELECT**: permette di leggere la risorsa
- **REFERENCES**: permette la definizione di vincoli di integrità referenziale verso la risorsa (può limitare la possibilità di modificare la risorsa)
- **USAGE**: permette l'utilizzo in una definizione (per esempio, di un dominio)

# Concessione e revoca di privilegi

- Concessione di privilegi:

**GRANT** *<Privileges | ALL PRIVILEGES >*  
**ON Resource TO Users [WITH GRANT OPTION]**

Es. **GRANT SELECT ON** Department **TO** Stefano

- Revoca di privilegi

**REVOKE Privileges ON Resource**  
**FROM Users [RESTRICT | CASCADE ]**



# Utenti e privilegi in MySQL

- Il server MySQL 5.5 memorizza l'informazione riguardo gli utenti nel database `mysql`
- L'amministratore del DBMS (utente predefinito `root`) è l'unico abilitato alla creazione di nuovi utenti (con indicazione di password e privilegi)
  - Es. creazione di un utente `pippo` in grado di connettersi da `localhost` al database `premiere` con password `mypwd` e privilegio di sola selezione
  - `sql> GRANT SELECT ON premiere.* TO pippo@localhost IDENTIFIED BY 'mypwd';`
  - L'esecuzione di questo comando comporta l'aggiornamento delle tabelle di `mysql`

# Utenti e privilegi in MySQL (II)

- Il server MySQL 5.5 consulta le tabelle di `mysql` ogni volta che un utente tenta di connettersi
  - Es. (cont.) L'utente `pip` può ora connettersi da `localhost` al database `premiere`
  - `shell> mysql -u pip -p premiere`
  - Il server riconosce l'utente `pip` a cui chiede di inserire la password per la successiva verifica
  - Se la fase di autenticazione ha successo, l'utente può interagire con il database `premiere` nella modalità concessa (accesso in interrogazione soltanto)
- Ogni modifica del profilo degli utenti (p.es. cambio di password) viene effettuata dall'amministratore di sistema mediante l'applicazione `mysqladmin.exe`

# Transazione

- *Insieme di operazioni da considerare indivisibile ("atomico"), corretto anche in presenza di concorrenza e con effetti definitivi*
- Proprietà ("acide"):
  - Atomicità
  - Consistenza
  - Isolamento
  - Durabilità (persistenza)

# Le transazioni sono ...

## ... atomiche

- La sequenza di operazioni sulla base di dati viene eseguita per intero o per niente:
  - trasferimento di fondi da un conto A ad un conto B: o si fanno il prelevamento da A e il versamento su B o nessuno dei due

## ... consistenti

- Al termine dell'esecuzione di una transazione, i vincoli di integrità debbono essere soddisfatti
- "Durante" l'esecuzione si possono verificare violazioni, ma se restano alla fine allora la transazione deve essere annullata per intero ("abortita")

# Le transazioni sono ...

## ... isolate

- L'effetto di transazioni concorrenti deve essere coerente (ad esempio "equivalente" all'esecuzione separata)
  - se due assegni emessi sullo stesso conto corrente vengono incassati contemporaneamente si deve evitare di trascurarne uno

## ... persistenti

- La conclusione positiva di una transazione corrisponde ad un impegno (in inglese *commit*) a mantenere traccia del risultato in modo definitivo, anche in presenza di guasti e di esecuzione concorrente

# Specifica di transazioni in SQL

- **BEGIN TRANSACTION**: specifica l'inizio della transazione (le operazioni non vengono eseguite sulla base di dati)
- **COMMIT WORK**: le operazioni specificate a partire dall'inizio della transazione vengono eseguite
- **ROLLBACK WORK**: si rinuncia all'esecuzione delle operazioni specificate dopo l'ultimo inizio di transazione
- **END TRANSACTION**: specifica la fine della transazione (le operazioni vengono eseguite sulla base di dati)

# Una transazione in SQL

**BEGIN TRANSACTION;**

**UPDATE** ContoCorrente

**SET** Saldo = Saldo - 10

**WHERE** NumeroConto = 12345 ;

**UPDATE** ContoCorrente

**SET** Saldo = Saldo + 10

**WHERE** NumeroConto = 55555 ;

**COMMIT WORK;**

**END TRANSACTION;**

# Esercitazione con MySQL

- Esercizi 1-7, 9 sul database Prodotti Premiere da cap. 6, pagg. 140-142 di Pratt;
- Esercizi a pagg. 175-177 di Atzeni et al.
- **N.B.** MySQL 5.5 supporta le viste e le transazioni ma non supporta né i vincoli di integrità generici né le asserzioni