

- SQL non è solo un linguaggio di interrogazione (Query Language), ma
- Un linguaggio per la definizione di basi di dati (Data-definition language (DDL))

CREATE SCHEMA Nome AUTHORIZATION Utente

CREATE TABLE o VIEW, con vincoli

CREATE INDEX

CREATE PROCEDURE

CREATE TRIGGER

- Un linguaggio per stabilire controlli sull'uso dei dati: GRANT
- Un linguaggio per modificare i dati.

DEFINIZIONE DI TABELLE: ESEMPIO

```
CREATE TABLE Impiegati
(
    Codice CHAR(8) NOT NULL,
    Nome CHAR(20),
    AnnoNascita INTEGER CHECK (AnnoNascita < 2000),
    Qualifica CHAR(20) DEFAULT 'Impiegato',
    Supervisore CHAR(8),
    PRIMARY KEY pk_impiegato (Codice),
    FOREIGN KEY fk_Impiegati (Supervisore)
    REFERENCES Impiegati )
```

```
CREATE TABLE FamiliariACarico
(
    Nome CHAR(20),
    AnnoNascita INTEGER,
    GradoParentela CHAR(10),
    CapoFamiglia CHAR(8)
    FOREIGN KEY fk_FamiliariACarico (CapoFamiglia)
    REFERENCES Impiegati )
```

- Ciò che si crea con un CREATE si può eliminare con il comando DROP o cambiare con il comando ALTER.

CREATE TABLE Nome

(Attributo Tipo [ValoreDefault] [VincoloAttributo]
{, Attributo Tipo [Default] [VincoloAttributo]}
{, VincoloTabella})

Default := DEFAULT {valore | null | username}

- Nuovi attributi si possono aggiungere con:

ALTER TABLE Nome ADD COLUMN NuovoAttributo Tipo

TABELLE INIZIALIZZATE E TABELLE CALCOLATE

- Tabelle inizializzate:

CREATE TABLE Nome EspressioneSELECT

CREATE TABLE Supervisor

SELECT Codice, Nome, Qualifica, Stipendio
FROM Impiegati
WHERE Supervisore IS NULL

- Tabelle calcolate (viste):

CREATE VIEW Nome [(Attributo {, Attributo})]
AS EspressioneSELECT [WITH CHECK OPTION];

CREATE VIEW Supervisor

AS SELECT Codice, Nome, Qualifica, Stipendio
FROM Impiegati
WHERE Supervisore IS NULL

- Vincoli su attributi

VincoloAttributo :=

[NOT NULL [UNIQUE]] | [CHECK (Condizione)]
[REFERENCES Tabella [(Attributo {, Attributo})]]

- Vincoli su tabella

VincoloTabella := UNIQUE (Attributo {, Attributo})
| CHECK (Condizione) |
| PRIMARY KEY [Nome] (Attributo {, Attributo})
| FOREIGN KEY [Nome] (Attributo {, Attributo})
REFERENCES Tabella [(Attributo {, Attributo})]
[ON DELETE {NO ACTION| CASCADE | SET NULL}]

ESEMPIO

6

- CREATE TABLE Impiegati
(Codice CHAR(8) NOT NULL,
Nome CHAR(20) NOT NULL,
AnnoNascita INTEGER NOT NULL,
Dipartimento CHAR(20),
Stipendio FLOAT NOT NULL,
Supervisore CHAR(8),
PRIMARY KEY pk_impiegato (Codice),
FOREIGN KEY fk_ Impiegati (Supervisore)
REFERENCES Impiegati
ON DELETE SET NULL
)

```

CREATE TABLE FamiliariACarico
( Nome CHAR(20) NOT NULL,
  AnnoNascita INTEGER NOT NULL,
  GradoParentela CHAR(10) NOT NULL,
  CapoFamiglia CHAR(8) NOT NULL,
  PRIMARY KEY pk_ FamiliariACarico (CapoFamiglia, Nome)
  FOREIGN KEY fk_ FamiliariACarico (CapoFamiglia)
  REFERENCES Impiegati
  ON DELETE CASCADE )

```

- CREATE TABLE Nazioni


```

      ( Codice CHAR(3) NOT NULL,
        Nome VARCHAR(50) NOT NULL,
        AnnoIndipendenza INTEGER,
        Popolazione INTEGER NOT NULL,
        Superficie FLOAT NOT NULL,
        Capitale INTEGER,
        PRIMARY KEY pk_nazione (Codice),
        FOREIGN KEY fk_citta (Capitale)
        REFERENCES Citta
        ON DELETE NO ACTION
      )
      
```
- ALTER TABLE Nazioni


```

      ADD CONSTRAINT (
        FOREIGN KEY fk_citta (Capitale)
        REFERENCES Citta
        ON DELETE NO ACTION
      )
      
```
- CREATE TABLE Citta


```

      ( Id INTEGER NOT NULL,
        Nome VARCHAR(50) NOT NULL,
        Popolazione INTEGER NOT NULL,
        Nazione CHAR(3) NOT NULL,
        PRIMARY KEY pk_citta (Id),
        FOREIGN KEY fk_nazione (Nazione)
        REFERENCES Nazioni
        ON DELETE CASCADE
      )
      
```
- ALTER TABLE Nazioni


```

      ADD CONSTRAINT (
        FOREIGN KEY fk_nazione (Nazione)
        REFERENCES Nazioni
        ON DELETE CASCADE
      )
      
```

```
• CREATE TABLE Città
( Id INTEGER NOT NULL,
  Nome VARCHAR(50) NOT NULL,
  Popolazione INTEGER NOT NULL,
  Nazione CHAR(3) NOT NULL,
  PRIMARY KEY pk_città (Id),
  FOREIGN KEY fk_città (Capitale)
  REFERENCES Città
  ON DELETE NO ACTION
)
```

CREATE PROCEDURE/FUNCTION

10

```
CREATE FUNCTION contaStudenti IS
DECLARE
  numStudenti INTEGER;
BEGIN
  SELECT COUNT(*) INTO numStudenti FROM STUDENTI;
  RETURN (numStudenti);
END
```

I TRIGGER

- I trigger si basano sul paradigma evento-condizione-azione (ECA):

CREATE TRIGGER Nome

PrimaODopoDi Evento {, Evento}

ON Tabella [WHEN Condizione]

[Granularità]

Azione

PrimaODopoDi := BEFORE | AFTER

Evento := INSERT | DELETE | UPDATE OF Attributi

Granularità := FOR EACH ROW | FOR EACH STATEMENT

ESEMPIO DI TRIGGER

CREATE TRIGGER ControlloStipendio

BEFORE INSERT ON Impiegati

DECLARE

StipendioMedio FLOAT

BEGIN

SELECT avg(Stipendio) INTO StipendioMedio

FROM Impiegati

WHERE Dipartimento = :new.Dipartimento;

IF :new.Stipendio > 2 * StipendioMedio

THEN RAISE_APPLICATION_ERROR(-2061, 'Stipendio alto')

END IF;

END;

- Proprietà essenziale dei trigger: terminazione
- Utilità dei trigger
 - Trattare vincoli non esprimibili nello schema
 - Attivare automaticamente azioni sulla base di dati quando si verificano certe condizioni

- Chi crea lo schema della BD è l'unico che può fare CREATE, ALTER e DROP
- Chi crea una tabella stabilisce i modi in cui altri possono farne uso:
 - GRANT Privilegi ON Oggetto TO Utenti [WITH GRANT OPTION]

- Tipi di privilegi:
 - SELECT: lettura di dati
 - INSERT [(Attributi)]: inserire record (con valori non nulli per gli attributi)
 - DELETE: cancellazione di record
 - UPDATE [(Attributi)]: modificare record (o solo gli attributi)
 - REFERENCES [(Attributi)]: definire chiavi esterne in altre tabelle che riferiscono gli attributi.
- WITH GRANT OPTION: si possono trasferire i privilegi ad altri utenti.

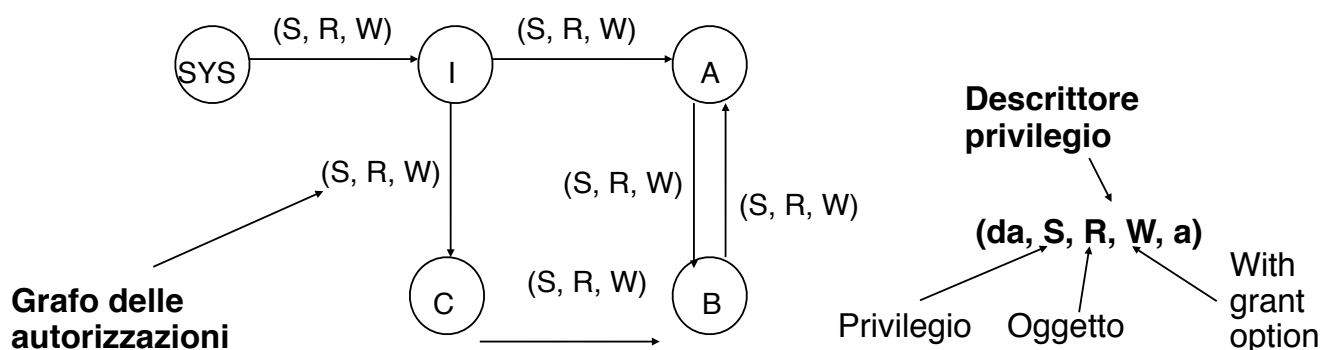
CONTROLLO DEGLI ACCESSI (cont.)

- Chi definisce una tabella o una VIEW ottiene automaticamente tutti i privilegi su di esse, ed è l'unico che può fare un DROP e può autorizzare altri ad usarla con GRANT.
- Nel caso di viste, il "creatore" ha i privilegi che ha sulle tabelle usate nella definizione.
- Le autorizzazioni si annullano con il comando:
 - REVOKE [GRANT OPTION FOR] Privilegi ON Oggetto FROM Utenti [CASCADE]
- Quando si toglie un privilegio a U, lo si toglie anche a tutti coloro che lo hanno avuto solo da U.

- GRANT INSERT, SELECT ON Esami TO Tizio .
- GRANT DELETE ON On Esami TO Capo WITH GRANT OPTION
 - Capo può cancellare record e autorizzare altri a farlo.
- GRANT UPDATE (voto) ON Esami TO Sicuro
 - Sicuro può modificare solo il voto degli esami.
- GRANT SELECT, INSERT ON VistaEsamiBD1 TO Albano
 - Albano può interrogare e modificare solo i suoi esami.

GRAFO DELLE AUTORIZZAZIONI

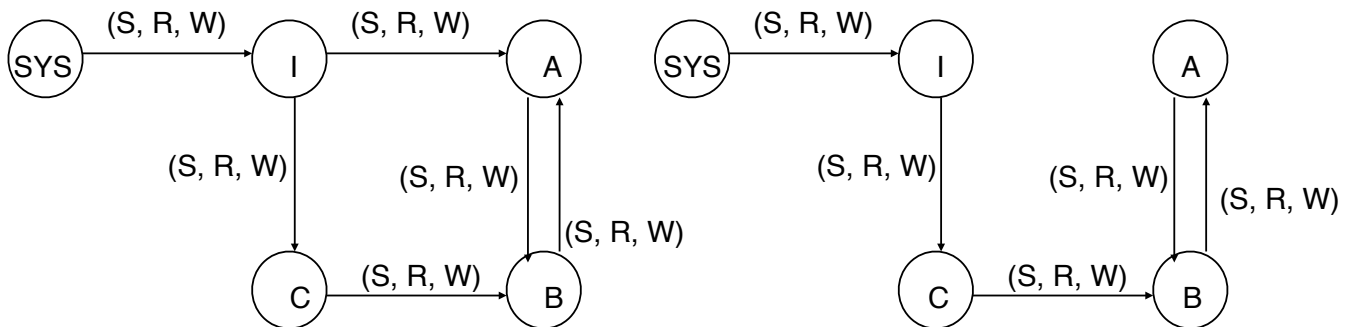
- L'utente I ha creato la tabella R e innesca la seguente successione di eventi:
 - I: GRANT SELECT ON R TO A WITH GRANT OPTION
 - A: GRANT SELECT ON R TO B WITH GRANT OPTION
 - B: GRANT SELECT ON R TO A WITH GRANT OPTION
 - I: GRANT SELECT ON R TO C WITH GRANT OPTION
 - C: GRANT SELECT ON R TO B WITH GRANT OPTION



- Se un nodo N ha un arco uscente con un privilegio, allora esiste un cammino da SYSTEM a N con ogni arco etichettato dallo stesso privilegio + WGO.
- Effetto del REVOKE, ad es.

I: REVOKE SELECT ON R FROM A CASCADE

- e poi I: REVOKE SELECT ON R FROM C CASCADE



CREAZIONE DI INDICI

- Cosa sono e a cosa servono
- Non è un comando standard dell'SQL e quindi ci sono differenze nei vari sistemi
 - CREATE INDEX NomeIdx ON Tabella(Attributi)
 - CREATE INDEX NomeIdx ON Tabella
WITH STRUCTURE = BTREE, KEY = (Attributi)
 - DROP INDEX NomeIdx

- Alcuni esempi di tabelle, delle quali si mostrano solo alcuni attributi, sono:

- Tabella delle password:

`PASSWORD(username, password)`

- Tabella delle basi di dati:

`SYSDB(dbname, creator, dbpath, remarks)`

- Tabella delle tabelle (type = view or table):

`SYSTABLES(name, creator, type, colcount, filename, remarks)`

- Alcuni esempi di tabelle, delle quali si mostrano solo alcuni attributi, sono:

- Tabella degli attributi:

`SYSCOLUMNS(name, tbname, tbcreator, colno, coltype, lenght, default, remarks)`

- Tabella degli indici:

`SYSINDEXES(name, tbname, creator, uniquerule, colcount)`

- e altre ancora sulle viste, vincoli, autorizzazioni, etc. (una decina).

- DDL consente la definizione di tabelle, viste e indici. Le tabelle si possono modificare aggiungendo o togliendo attributi e vincoli.
- Le viste si possono interrogare come ogni altra tabella, ma in generale non consentono modifiche dei dati.
- I comandi *GRANT* / *REVOKE* + viste offrono ampie possibilità di controllo degli usi dei dati.

- SQL consente di dichiarare molti tipi di vincoli, oltre a quelli fondamentali di chiave e referenziale.
- Oltre alle tabelle fanno parte dello schema le procedure e i trigger.
- La padronanza di tutti questi meccanismi -- e di altri che riguardano aspetti fisici, affidabilità, sicurezza -- richiede una professionalità specifica (DBA).