DATABASE IMPIEGATIDIPARTIMENTI

TESTO

Esempio tratto dal libro di testo.

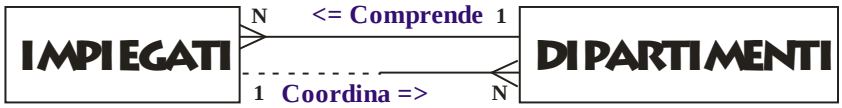
Un'azienda è articolata in un certo numero di dipartimenti. I dipendenti sono assegnati ai diversi dipartimenti che hanno a capo un manager responsabile della gestione.

Nell'esempio saranno via via sviluppate le varie funzionalità del linguaggio SQL.

ANALISI DEL TESTO

Vista la natura dell'esempio, ove si ammette che ogni impiegato possa coordinare uno o anche più dipartimenti (mentre ogni dipartimento dovrà essere coordinato da un unico manager), ridurremo al minimo indispensabile Entità, Attributi e Relazioni.

MODELLO E/R



MODELLO LOGICO

TABELLA IMPIEGATI

ATTRIBUTO	TIPO	LEN	FORMATO	KEY	Ind	DESCRIZIONE
IdImpiegato	Intero	-		PK	si	
Cognome	Testo	30			si	
Nome	Testo	20			si	
Residenza	Testo	30				Solo città
Stipendio	Reale	-	Valuta €			Stipendio annuo
IdDipartimento	Testo	5	Codificato	FK	si	Dipartimenti.IdDipartimento

TABELLA DIPARTIMENTI

ATTRIBUTO	TIPO	LEN	FORMATO	KEY	Ind	DESCRIZIONE
IdDipartimento	Testo	5	Codificato	PK	si	Allegato Codice Dipartimento
Descrizione	Testo	30				
Sede	Testo	30				Solo città
Manager	Intero	-		FK	si	Impiegati.IdImpiegato

ALLEGATI

CODICE DIPARTIMENTO

Valido per *Impiegati* e *Dipartimenti*.

IdDipartimento	Significato
Amm	Amministrazione
Direz	Direzione
Mag	Magazzino
Mkt	Marketing
Prod	Produzione
R&S	Ricerca e Sviluppo

Definizione Tabelle e Manipolazione Dati

```
CREATE TABLE Impiegati
  (IdImpiegato        Integer Primary Key,
   Nome               Char(20) not null,
   Cognome            Char(30) not null,
   Residenza          Char(30),
   Stipendio          Decimal(9,2) default 0,
   IdDipartimento     Char(5) references Dipartimenti(IdDipartimento),
   Unique(Nome, Cognome, IdImpiegato));

CREATE TABLE Dipartimenti
  (IdDipartimento     Char(5),
   Descrizione        Char(30),
   Sede               Char(30),
   Manager            Integer,
   Primary Key(IdDipartimento),
   Unique(Descrizione),
   Foreign Key(Manager) references Impiegati.IdImpiegato,
     on delete set null,
     on update cascade);
```

DEFINIZIONE TABELLE	
CREATE UNIQUE ImpiegatiIndex ON Impiegati(Nome, Cognome);	CREATE UNIQUE DipartimentiIndex ON Dipartimenti(Descrizione);
DROP TABLE Impiegati;	DROP INDEX DipartimentiIndex ON Dipartimenti;
ALTER TABLE Impiegati ADD DataNascita Date;	ALTER TABLE Dipartimenti DROP Sede;
MANIPOLAZIONE DATI	
INSERT INTO Impiegati (IdImpiegato, Nome, Cognome, Residenza, Stipendio, IdDipartimento) VALUES(20, ‘Dario’, ‘Rossi’, ‘Milano’, 35000, ‘Mag’);	
UPDATE Impiegati SET IdDipartimento = ‘Prod’, Stipendio = 38000 Where IdImpiegato = 20;	UPDATE Impiegati SET Stipendio = Stipendio * 1.05 Where IdDipartimento = ‘Prod’;
DELETE FROM Impiegati WHERE IdImpiegato = 5;	DELETE FROM Impiegati WHERE IdDipartimento = ‘Prod’;

Interrogazioni

SELEZIONE - PROIEZIONE

IMPIEGATIDIPPROD

```
SELECT Cognome, Nome, Residenza
FROM Impiegati
WHERE IdDipartimento = 'Prod';
```

IMPIEGATIDIPPRODROMA

```
SELECT Cognome, Nome, Residenza
FROM Impiegati
WHERE IdDipartimento = 'Prod'    AND Residenza = 'Roma';
```

IMPIEGATIROMA

```
SELECT *
FROM Impiegati
WHERE Residenza = 'Roma';
```

AUMENTOSTIPENDI

```
SELECT Cognome, Nome, Stipendio, Stipendio*1.05 AS NuovoStipendio
FROM Impiegati
ORDER BY Cognome, Nome;
```

IMPIEGATIRESIDENZE

```
SELECT DISTINCT Residenza
FROM Impiegati
ORDER BY Residenza;
```

IMPIEGATISENZADIP

```
SELECT IdImpiegato AS Matricola, Cognome, Nome, Residenza
FROM Impiegati
WHERE IdDipartimento IS NULL;
```

STIPENDIDA55000

```
SELECT Cognome, Nome, Stipendio
FROM Impiegati
WHERE Stipendio >= 55000
ORDER BY Stipendio;
```

STIPENDIDAX

```
SELECT Cognome, Nome, Stipendio
FROM Impiegati
WHERE Stipendio >= [ Immettere Stipendio Minimo: ]
ORDER BY Stipendio;
```

CONGIUNZIONE (JOIN)

$Impiegati_{FK} \triangleright\!\triangleleft Dipartimenti_{PK}$

Cong di Imp e Dip su IdDip

Cong di Imp.IdDip su Dip.IdDip

ELENCOIMPIEGATICONDATIDIP

```
SELECT *
FROM Impiegati, Dipartimenti
WHERE Impiegati.IdDipartimento = Dipartimenti.IdDipartimento;
```

Versione Access

```
SELECT Impiegati.*, Dipartimenti.*
FROM Impiegati INNER JOIN Dipartimenti ON
        Impiegati.IdDipartimento = Dipartimenti.IdDipartimento;
```

Raggruppamento per Dipartimento-Cognome-Nome – Eliminazione colonna ridondante Dipartimenti.IdDipartimento – Ridenominazione di Impiegati.IdDipartimento come Dipartimento

```
SELECT Impiegati.IdImpiegato, Impiegati.Cognome, Impiegati.Nome, Impiegati.Residenza, Impiegati.Stipendio,
Impiegati.IdDipartimento AS Dipartimento, Dipartimenti.Descrizione, Dipartimenti.Sede, Dipartimenti.Manager
FROM Dipartimenti INNER JOIN Impiegati ON Dipartimenti.IdDipartimento=Impiegati.IdDipartimento
ORDER BY Impiegati.IdDipartimento, Impiegati.Cognome, Impiegati.Nome;
```

σ_P D T1=Sel di D per Sede='Roma'

$T1_{FK} \triangleright\!\triangleleft I_{PK}$

T2=Cong di T1 e I su Id.Dip

Π_L T2

Proiez di T2 su Cogn,Nome,Descr

ELENCOIMPIEGATIDIPROMACONDESCRIZIONEDIP

```
SELECT Cognome, Nome, Descrizione AS Dipartimento
FROM Impiegati, Dipartimenti
WHERE Impiegati.IdDipartimento = Dipartimenti.IdDipartimento AND
        Sede = 'Roma';
```

Versione Access

```
SELECT I.Cognome, I.Nome, D.Descrizione AS Dipartimento
FROM Dipartimenti AS D INNER JOIN Impiegati AS I ON
        D.IdDipartimento = I.IdDipartimento
WHERE D.Sede = 'Roma';
```

FUNZIONI DI AGGREGAZIONE (COUNT - SUM - AVG - MIN - MAX)

IMPIEGATITUPLE

```
SELECT COUNT(*) AS Records
FROM Impiegati;
```

NUMEROIMPIEGATIDIP

```
SELECT COUNT(IdDipartimento) AS Impiegati_Dipartimento
FROM Impiegati
WHERE IdDipartimento = [ Immetti Codice Dipartimento ]
```

NUMEROIMPIEGATIRESIDENZA

```
SELECT COUNT(*) AS Impiegati
FROM Impiegati
WHERE Residenza = [ Input Residenza ];
```

NUMERORESIDENZEDIPPROD

```
SELECT COUNT(DISTINCT Residenza) AS Numero_Provenienze
FROM Impiegati
WHERE IdDipartimento = 'Prod';
```

TOTSTIPENDIDIPMKT

```
SELECT SUM(Stipendio) AS TotStipMkt
FROM Impiegati
WHERE IdDipartimento = 'Mkt';
```

MEDIASTIPENDIDIPSEDETORINO

```
SELECT AVG(Stipendio) AS Media
FROM Impiegati, Dipartimenti
WHERE Impiegati.IdDipartimento = Dipartimenti.IdDipartimento AND
      Sede = 'Torino';
```

Versione Access

```
SELECT AVG(Stipendio) AS Media
FROM Impiegati INNER JOIN Dipartimenti ON
     Impiegati.IdDipartimento = Dipartimenti.IdDipartimento
WHERE Sede = 'Torino';
```

STIPENDIOMINMAX

```
SELECT MIN(Stipendio) AS Minimo,
       MAX(Stipendio) AS Massimo
FROM Impiegati;
```

COGNOMEMINMAX

```
SELECT MIN(Cognome) AS Primo, MAX(Cognome) AS Ultimo
FROM Impiegati;
```

ORDINAMENTO E RAGGRUPPAMENTO (ORDER BY - GROUP BY)

ELENCOSTIPENDIDECRESCENTE

```
SELECT Cognome, Stipendio
FROM Impiegati
ORDER BY Stipendio DESC, Cognome;
```

ELENCOALFABETICOIMPIEGATI

```
SELECT Cognome, Nome, IdImpiegato AS Matricola
FROM Impiegati
ORDER BY Cognome, Nome;
```

ELENCODIPTOTSTIPENDINUMIMPIEGATI - 1

```
SELECT IdDipartimento, SUM(Stipendio) AS TotaleStipendi,
       COUNT(IdImpiegato) AS NumeroImpiegati
FROM Impiegati
GROUP BY IdDipartimento;
```

Con la **Descrizione** del Dipartimento invece che la **FK** Impiegati.IdDipartimento

ELENCODIPTOTSTIPENDINUMIMPIEGATI - 2

```
SELECT Descrizione, SUM(Stipendio) AS TotaleStipendi,
       COUNT(*) AS NumeroImpiegati
FROM Impiegati, Dipartimenti
WHERE Impiegati.IdDipartimento = Dipartimenti.IdDipartimento
GROUP BY Descrizione;
```

Versione Access

```
SELECT Descrizione, SUM(Stipendio) AS TotaleStipendi,
       COUNT(*) AS NumeroImpiegati
FROM Impiegati INNER JOIN Dipartimenti ON
       Impiegati.IdDipartimento = Dipartimenti.IdDipartimento
GROUP BY Descrizione;
```

CONDIZIONI SUI RAGGRUPPAMENTI (HAVING)	
Se per le funzioni SUM e COUNT non si applica la clausola AS le relative titolazioni di colonna vengono assegnate dal sistema	**ELENCODIPTOTSTIPENDIMIN3IMPIEGATI** SELECT IdDipartimento, SUM(Stipendio), COUNT(IdImpiegato) FROM Impiegati GROUP BY IdDipartimento HAVING COUNT(IdImpiegato) > 2;
Se per le funzioni SUM e COUNT non si applica la clausola AS le relative titolazioni di colonna vengono assegnate dal sistema	**ELENCODIPTOTSTIPENDIMIN3IMPIEGATISEDETORINO** SELECT Descrizione, SUM(Stipendio), COUNT(*) FROM Impiegati, Dipartimenti WHERE Impiegati.IdDipartimento = Dipartimenti.IdDipartimento AND Sede = 'Torino' GROUP BY Descrizione HAVING COUNT(*) > 2;
CONDIZIONI DI RICERCA (BETWEEN - IN - LIKE - IS NULL)	
Equivale a Stipendio >= 3000 AND Stipendio <= 4500 Usare NOT per invertire	**ELENCOSTIPENDIRANGE** SELECT Cognome, Nome, Stipendio FROM Impiegati WHERE Stipendio BETWEEN 3000 AND 4500;
Equivale a Residenza = 'Roma' OR Residenza = 'Torino' Usare NOT per invertire	**ELENCORESIDENZESCELTE** SELECT * FROM Impiegati WHERE Residenza IN ('Roma', 'Torino');
Caratteri Jolly (Metacaratteri): **_ (underline)** = carattere Generico **%** = stringa generica (0 o più caratteri)	**ELENCOCOGNOMIINIZIOCARGENERICO+OSS+STRGENERICA** SELECT Cognome, Nome, IdDipartimento FROM Impiegati WHERE Cognome LIKE '_oss%';
Il **carattere di escape** definito con la dichiarazione ESCAPE va (opportunamente) scelto dall'utente	**ELENCOCOGNOMICONUNDERSCORE** SELECT Cognome, Nome, IdDipartimento FROM Impiegati WHERE Cognome LIKE 'De$_San%' ESCAPE '$';
IS è un operatore utilizzabile solo col "valore" NULL. IS NULL è un predicato Usare NOT per invertire	**CHECKDESCRIZIONEDIP** SELECT IdDipartimento, Descrizione FROM Dipartimenti WHERE Descrizione IS NOT NULL;
VISTE LOGICHE (VIEW)	
Access Comando non disponibile Si ricorre ad altre tecniche	**VISTADIPTOTSTIPENDINUMIMPIEGATI** CREATE VIEW TotStipImpxDip (Dip, TotStip, NumImp) AS SELECT IdDipartimento, SUM(Stipendio), COUNT(*) FROM Impiegati GROUP BY IdDipartimento;

SICUREZZA (GRANT - REVOKE)

REVOCAUPDATE	**PERMESSOUPDATESELETTIVO**
REVOKE UPDATE ON Impiegati FROM User12;	GRANT UPDATE (Residenza, Stipendio) ON Impiegati FROM User08, User12;

Diritti di Accesso	**Significato**
ALTER	Update/Delete colonne – Update tipi di dati
DELETE	Delete righe
INDEX	Creare indici
INSERT	Inserire righe
SELECT	Ricercare dati
UPDATE	Update valori
ALL	Tutti i permessi

TRIGGER O REGOLE EVENTO-CONDIZIONE-AZIONE (TRIGGER)

Non disponibile in Access	**CHECKSTIPENDINODIMINUZIONI**
Definizione Trigger	CREATE TRIGGER CheckStipendi
Evento Update su Impiegati	AFTER UPDATE OF Stipendio ON Impiegati
	REFERENCING NEW ROW AS NuovoS OLD ROW AS VecchioS
Condizione attivazione Trigger	WHEN NuovoS.Stipendio < VecchioS.Stipendio
	FOR EACH ROW

Azione se Condizione vera	BEGIN
	UPDATE Impiegati
	SET Stipendio = VecchioS.Stipendio
	WHERE IdImpiegato = NuovoS.IdImpiegato
	END;

Non disponibile in Access	**CHECKDIPMAX99IMPIEGATI**
Definizione Trigger	CREATE TRIGGER CheckDipartimenti
Evento Insert su Impiegati	AFTER INSERT ON Impiegati
	REFERENCING NEW AS NuovoI
Condizione attivazione Trigger	WHEN (100 < (SELECT COUNT(*) FROM Impiegati WHERE IdDipartimento = NuovoI.IdDipartimento))
	FOR EACH ROW

Azione se Condizione vera	BEGIN
	DELETE FROM Impiegati
	WHERE IdImpiegato = NuovoI.IdImpiegato
	END;