Esempi di Queries(interrogazioni):

SELECT Nome, Cognome FROM Persone WHERE id = 1

SELECT Nome, Cognome FROM Persone WHERE Nome = “Luigi”

SELECT ***** FROM Persone WHERE Nome = “Luigi”

*(* = visualizza tutti i campi ovvero tutte le colonne)*

OPERATORI LOGICI: AND, OR, NOT

AND = … E … → tutte le condizioni devono essere soddisfatte

SELECT ***** FROM Persone WHERE (Nome = ‘Mario’) **AND** (Cognome= ‘Rossi’)

Restituisce i dati di coloro che si chiamano Mario Rossi(potrebbero esservi degli omonimi) ovvero tutte le righe che hanno come valori dei campi Nome e Cognome rispettivamente Mario e Rossi

OR = … O … → almeno una delle condizioni deve essere soddisfatta

SELECT ***** FROM Persone WHERE (Nome = ‘Mario’) **OR** (Nome= ‘Luigi’)

Restituisce i dati di coloro che si chiamano Mario o Luigi

NOT = NON … → non deve essere soddisfatta una certa condizione

SELECT ***** FROM Persone WHERE **NOT**(Nome = ‘Mario’)

Restituisce i dati di coloro che non si chiamano Mario (TUTTE LE RIGHE ECCETTO quelle relative alle persone di nome Mario)

Gli operatori possono essere **usati anche insieme** come si può vedere dal seguente esempio:

SELECT ***** FROM Persone WHERE (**NOT**(Nome = ‘Mario’) **AND** (Cognome= ‘Rossi’)) **OR** (Nome =’Luigi’)

In questo caso di devono **guardare le parentesi**, nell’esempio riportato l’OR “lega” le due condizioni:

A: (**NOT**(Nome = ‘Mario’) **AND** (Cognome= ‘Rossi’)) = tutti gli utenti aventi cognome = Rossi che non si chiamano Mario

B: (Nome =’Luigi’)

Quindi vengono prese tutte le righe che soddisfano la prima o la seconda condizione.

CONFRONTO PARZIALE STRINGHE: LIKE e Caratteri JOLLY

Per il confronto parziale di stringhe vengono utilizzati **LIKE** e dei **caratteri speciali** . Noi ne vedremo solo uno: *

*** = sequenza di caratteri alfanumerici**

Esempi:

ca* = tutte le stringhe che **iniziano** per "ca"

ca = tutte le stringhe che **contengono** "ca"

*ca = tutte le stringhe che **finiscono** "ca"

Esempio di interrogazione al DB: SELECT * FROM Persone WHERE (Nome **LIKE "ma*"**)

Restituisce i dati delle persone il cui Nome inizia per ma(non importa se i caratteri sono maiuscoli o minuscoli). Ad es. potrebbe restituire i dati di Mario ,Manuela, Maria, Massimo,…

FUNZIONI DI AGGREGAZIONE

Servono ad elaborare i dati di un'intera colonna(o di una parte degli elementi della colonna se utilizziamo WHERE per selezionare solo alcune righe)

Tali funzioni sono:

1. **COUNT**: **numero di elementi** della colonna specificata
2. **MIN**: valore **minore**, tra quelli della colonna
3. **MAX**: valore **maggiore**, tra quelli della colonna
4. **SUM**: **somma** dei valori della colonna
5. **AVG**: **media aritmetica**(average)dei valori della colonna(= Sum(colonna) / Count(colonna))

Esempi:

SELECT **COUNT(id)** FROM Persone ;

restituisce il numero di valori nella colonna id e quindi il numero di elementi della tabella Persone

Avrei potuto utilizzare anche la seguente query: SELECT **COUNT(*)** FROM Persone ;

SELECT **MIN(stipendio)** FROM Dipendenti ; restituisce il valore dello stipendio più basso

SELECT **MAX(stipendio)** FROM Dipendenti ; restituisce il valore dello stipendio più alto

SELECT **AVG(stipendio)** FROM Dipendenti ; restituisce lo stipendio medio

SELECT **SUM(importo)** FROM Articoli_acquistati WHERE data=#20/01/2013#;

restituisce l'importo totale di tutti gli articoli acquistati nel giorno specificato(la tabella articoli avrà come colonne: id_articolo(chiave primaria), importo, data, descrizione, tipo,…)

ORDINAMENTO

Uso la clausola ORDER BY [ASC|DESC] dove **ASC** specifica che voglio che i risultati della query siano ordinate in ordine **crescente** rispetto alla colonna considerata (dal più piccolo al più grande), **DESC → decrescente**.

SELECT ***** FROM Dipendenti ORDER BY stipendio DESC;

I risultati(tutte le righe della tabella Dipendenti) verranno ordinati in base al valore del campo stipendio (dal più grande al più piccolo)

In assenza di opzioni viene sottinteso ASC.

LIMITARE RISULTATI

Per limitare i risultati ai primi N in Access si usa TOP.

SELECT * from Persone **TOP 10** → prime 10 righe della tabella

GIUNZIONE TRA DUE TABELLE

In genere le tabelle che vengono utilizzate sono collegate tra di loro…

Esempio:

Voglio registrare tutte le **operazioni** (versamento/prelievo) effettuate sui **conti correnti** dei **clienti** di una banca:

Avrò almeno 3 tabelle: **Clienti**, **ContiCorrenti** e **Operazioni**

(per semplicità supporremo che ogni conto possa avere un solo intestatario)

1)**Clienti** : dati degli utenti(id_cliente(chiave primaria), nome, cognome, tel, via, dataNascita, codicefiscale…)

2)**ContiCorrenti** : dati dei conti correnti (id_conto(chiave primaria),**id_intestatario**, data_ora_apertura, data_ora_chiusura, …)

3)**Operazioni**: dati relativi alle operazioni (id_operazione(chiave primaria), **id_cc**, importo(negativo in caso di prelievo), data_ora,…)

E' possibile notare che i dati presenti nelle tabelle sono collegati attraverso i campi

id_intestatario → id_cliente

id_cc → id_conto

i conti correnti sono associati in questo modo ai leggittimi possessori, mentre le operazioni ai conti correnti sui quali sono state effettuate

Tali campi sono dei **RIFERIMENTI alle chiavi primarie** di altre tabelle e vengono detti **CHIAVI ESTERNE .**

Supponiamo di aver creato le suddette tabelle con Access e che vogliamo sapere chi sono gli intestari dei vari conti correnti… è possibile ottenerlo attraverso la seguente interrogazione:

SELECT * FROM Clienti,ContiCorrenti WHERE (Clienti.id_cliente = ContiCorrenti.id_intestatario)

Ecco come funziona la query:

1) SELECT * FROM Clienti,ContiCorrenti restituisce **tutte le possibili combinazioni cliente → conto corrente**(con tutti i campi perché c'è l'asterisco), se i clienti sono 3 e i conti correnti 4, avrò 3x4= 12 possibili combinazioni in cui ognuno dei clienti è associato a tutti e 4 i conti(anche quelli non suoi!!!)

2) Questo però non è quello che vogliamo, dobbiamo "filtrare" i risultati ottenuti in maniera che ogni cliente sia associato SOLO AI PROPRI CONTI CORRENTI(se ne ha più di uno) ed è a questo che serve il WHERE!! Infatti nel WHERE viene controllato se l'intestatario del conto(CHIAVE ESTERNA) corrisponde all'ID del CLIENTE ovvero se il cliente è il proprietario del conto oppure no. Solo le righe per le quali si verifica questa corrispondenza saranno restituite alla fine

   La GIUNZIONE è proprio questa operazione di associazione tra i dati in cui si verifica che CHIAVE_ESTERNA_Tabella_2 = CHIAVE_PRIMARIA_Tabella_1

   Ovviamente il tutto si può estendere a più tabelle purché si uguaglino le coppie di chiavi giuste.

PROBLEMI:

1) Calcolare il **saldo del conto** avente **id_conto = 2 ?**

Saldo conto = Somma degli importi(positivi in caso di versamenti e negativi per i prelievi) delle operazioni effettuate

SELECT SUM(importo) FROM Operazioni WHERE (Operazioni.id_cc = 2)

2) Trovare i **conti correnti dell'utente** avente **codice fiscale = CR… ?**

SELECT id_conto FROM Clienti,ContiCorrenti WHERE (Clienti.id_cliente = ContiCorrenti.id_intestatario) AND (Clienti.codicefiscale = "CR…")

3) Trovare **le ultime 10 operazioni** effettuate dall'utente avente **codice fiscale = CR… ?**

SELECT Operazioni.* FROM Clienti,ContiCorrenti,Operazioni WHERE (Clienti.id_cliente = ContiCorrenti.id_intestatario) AND (Operazioni.id_cc = ContiCorrenti.id_conto) AND (Cliente.codicefiscale = "CR…") ORDER BY Operazioni.data_ora DESC TOP 10

Gli istanti di tempo(data, ora o entrambi) possono essere ordinati e confrontati come se fossero numeri interi…(in effetti è questo il modo in cui internamente sono memorizzati).

Per intenderci:

20/02/2014 < 21/02/2014 e 20/02/2014 13:20 < 20/02/2014 13:25