Università degli Studi de L'Aquila
A. A. 2001-2002

Prof. Giuseppe Della Penna

PROGETTO DI LABORATORIO DI BASI DI DATI

"Software House"

GRUPPO 19 – CANALE A

A cura di:

Giovanni Gasparri
Paolo Palleschi
Carmine Di Natale
Danilo Lombardi

e-mail di riferimento: giovanni@gasparrisoft.com

INDICE GENERALE

I – INTRODUZIONE

1.1 - Formulazione del problema

Si vuole realizzare una base di dati per la gestione di una società di sviluppo software.

La società porta avanti una serie di progetti. Ciascun progetto ha un nome, una descrizione, una data di inizio e una data prevista per la consegna. Un progetto prevede lo sviluppo di diverse applicazioni software, ciascuna classificata tramite il suo "ruolo" (interfaccia, base di dati, ecc…).

Per ogni applicazione abbiamo un nome, la data in cui è iniziato il suo sviluppo, la data in cui è iniziato il debugging e la data del rilascio, il linguaggio con cui viene sviluppata e le qualifiche richieste per la sua realizzazione (si veda più avanti).

E' inoltre disponibile una lista di tutto il personale che è assegnato allo sviluppo dell'applicazione. Poiché, come noto, il software necessita di continui ritocchi, per ogni applicazione viene memorizzata una lista degli interventi di aggiornamento/correzione, con una data, il personale adibito alle modifiche, e una descrizione dell'intervento effettuato.

La società dispone di un archivio personale. Il personale può essere regolarmente assunto o essere registrato come "collaboratore occasionale". In ogni caso, siamo interessati a conservare i dati anagrafici completi di ciascun membro del personale. Per gli impiegati, vengono anche registrati il numero di matricola, la data di assunzione, lo stipendio attuale. Per i collaboratori, viene memorizzata una lista di tutti i periodi in cui hanno lavorato per la società, con il compenso ricevuto e il progetto al quale hanno collaborato.

E' importante conoscere quali sono le "qualifiche" di ciascun membro del personale. Le qualifiche sono inserite in una lista predefinita (ad esempio "Programmatore C", "Programmatore Java", esperto di un determinato settore di sviluppo, ecc.), e ad ogni persona registrata nell'archivio personale sono associate le sue qualifiche.

Periodicamente, la società organizza corsi di aggiornamento per gli impiegati, tenuti da collaboratori esterni. Ciascun corso fornisce a chi lo frequenta una serie di qualifiche.

Ogni corso avrà un nome, un numero di ore di lezione, un docente (membro del personale classificato come collaboratore), il periodo nel quale si tiene. Nella base di dati sono registrati tutti gli impiegati che hanno partecipato a ciascun corso, e il periodo in cui lo hanno seguito.

1.2 - Obiettivi

1. Definire formalmente i requisiti.
2. Progettare concettualmente lo schema utilizzando il modello Entity-Relationship.
3. Definire lo schema E-R ristrutturato e documentare dettagliatamente i vincoli di integrità.
4. Tradurre il progetto nel modello relazionale.
5. Implementare il progetto su un DBMS di propria scelta.

1.3 - Svolgimento del Progetto

- Per la progettazione concettuale è stata adottata principalmente la strategia **Inside-Out**;
- Per l'implementazione del progetto è stato scelto il DBMS **PostgreSQL** versione 7.1.3 per **Mac OS X 10.1**;
- Per la realizzazione del frontend è stato scelto l'ambiente di sviluppo **Real Basic** vers. **3.2.1 Professional** per Mac OS X/Classic;
- Per l'implementazione dei Vincoli di integrità sono state utilizzate **esclusivamente** funzioni di controllo a livello **Back-End**;
- Per la stesura delle funzioni e delle procedure è stato utilizzato il linguaggio **PLPGSQL**;

Sono stati scelti questi strumenti perché:

- La piattaforma Mac OS X, essendo nata da poco, è inesplorata per quanto riguarda le applicazioni **UNIX-BASED**;
- Essendo quasi inesistente la **documentazione** sull'interazione tra PostgreSQL e Mac OS X, si vogliono rendere disponibili ad altri studenti universitari preziose informazioni sull'implementazione di SQL in ambiente Carbon;
- Il frontend realizzato con Real Basic garantisce l'**originalità** del lavoro e l'impegno affrontato per la realizzazione di questo progetto;
- Si è avuta la possibilità di conoscere ed utilizzare il **terminale Darwin** (CORE UNIX) per compilare, installare, eseguire ed interrogare il DBMS;

Non è stato ritenuto opportuno utilizzare linguaggi di scripting server-side per interfacciarsi alla Base Dati.

II – ANALISI DEI REQUISITI

2.1 - Glossario

TERMINE	DESCRIZIONE	SINONIMI
Lavoratore	Membro del personale della società indistintamente tra collaboratore o impiegato.	Persone
Impiegato	Lavoratore regolarmente assunto dalla società.	Dipendente
Collaboratore	Lavoratore che ha avuto rapporti occasionali con la società.	Collaboratore esterno
Ruolo	Funzionalità specifica di un'applicazione.	
Qualifiche	Insieme di titoli che specializzano un lavoratore in un determinato settore.	
Intervento	Intervento di aggiornamento e/o correzione di un'applicazione.	Modifica, correzione, aggiornamento
Docente	Collaboratore abilitato ad insegnare ai corsi di aggiornamento.	Insegnante
Serie di Qualifiche	Almeno una qualifica.	
Progetto attivo	Progetto non ancora rilasciato.	
Lavoratore Libero	Lavoratore non assegnato ad alcun progetto attivo.	

2.2 - Strutturazione dei requisiti

Espressioni di Carattere Generale
Si vuole realizzare una base di dati per la gestione di una società di sviluppo software.

Espressioni relative ai PROGETTI
La società porta avanti una serie di progetti. Ciascun progetto ha un nome, una descrizione, una data di inizio e una data prevista per la consegna. Ciascun progetto prevede lo sviluppo di diverse applicazioni.

Espressioni relative alle APPLICAZIONI

Ogni applicazione è classificata tramite il ruolo. Ogni applicazione ha un nome, la data in cui è iniziato lo sviluppo, la data in cui è iniziato il debugging, la data di rilascio, il linguaggio con cui viene sviluppata, le qualifiche richieste per la realizzazione.

E' disponibile una lista dei lavoratori assegnati allo sviluppo di un'applicazione. Per ogni applicazione viene memorizzata una lista di interventi con una data, i lavoratori addetti ad una descrizione dell'intervento.

Espressioni relative ai LAVORATORI

Per ogni lavoratore vogliamo conservare i dati anagrafici completi. Per gli impiegati registriamo anche il numero di matricola, la data di assunzione, lo stipendio attuale. Per i collaboratori registriamo una lista di tutti i periodi in cui hanno lavorato per la società, con il compenso ricevuto e il progetto al quale hanno collaborato.

Ad ogni lavoratore sono associate le sue qualifiche (selezionabili da un elenco predefinito).

Per ogni lavoratore vogliamo conoscere le sue qualifiche, gli interventi ed i progetti nei quali ha lavorato.

Espressioni relative ai CORSI

La società organizza corsi di aggiornamento per gli impiegati tenuti da collaboratori. Ciascun corso fornisce agli impiegati che lo frequentano una serie di qualifiche. Ogni corso ha un nome, un numero di ore di lezione, un docente, il periodo nel quale si tiene.

Nel database dono registrati tutti gli impiegati che hanno partecipato a ciascun corso, ed il periodo in cui lo hanno seguito.

2.3 - Operazioni previste sui dati

Progetti

NOME	OPERAZIONE
Op. 1	Inserimento di un nuovo progetto
Op. 2	Modifica di un progetto
Op. 3	Eliminazione di un progetto
Op. 4	Inserimento delle applicazioni software legate al progetto
Op. 5	Ricerca dei progetti che hanno superato il periodo di consegna previsto
Op. 6	Elenco dei linguaggi utilizzati nello sviluppo di un progetto
Op. 7	Elenco delle qualifiche necessarie allo sviluppo di un progetto
Op. 8	Verifica della presenza, tra il personale assegnato ad un progetto, di tutte le qualifiche necessarie
Op. 9	Ricerca dei progetti correntemente in sviluppo e del loro stato di avanzamento (numero delle applicazioni completate)

Applicazioni

NOME	OPERAZIONE
Op. 10	Inserimento di una nuova applicazione
Op. 11	Modifica dei dati di un'applicazione
Op. 12	Assegnazione di personale allo sviluppo di un'applicazione
Op. 13	Verifica della presenza, tra il personale assegnato ad un'applicazione, di tutte le qualifiche necessarie al completamento dello sviluppo
Op. 14	Registrazione degli interventi

Personale

NOME	OPERAZIONE
Op. 15	Inserimento di un nuovo impiegato
Op. 16	Modifica/Eliminazione di un impiegato
Op. 17	Inserimento dei dati riguardanti i collaboratori (periodo, progetto, compenso)
Op. 18	Ricerca di tutto il personale con particolari qualifiche
Op. 19	Determinazione di tutti i corsi di aggiornamento frequentati da un impiegato
Op. 20	Ricerca di tutto il personale attualmente libero (non assegnato ad alcuna applicazione attiva)

Qualifiche

NOME	OPERAZIONE
Op. 21	Inserimento di una nuova qualifica
Op. 22	Modifica/Eliminazione di una qualifica

Corsi

NOME	OPERAZIONE
Op. 23	Inserimento di un nuovo corso
Op. 24	Modifica/Eliminazione di un corso
Op. 25	Assegnazione ad un corso delle qualifiche da esso derivanti
Op. 26	Aggiunta di partecipanti al corso
Op. 27	Aggiornamento delle qualifiche degli impiegati in base ai corsi frequentati

Operazioni Aggiuntive

NOME	OPERAZIONE
Op. 28	Ricerca di tutte le qualifiche di un lavoratore
Op. 29	Ricerca di tutte le applicazioni di un progetto
Op. 30	Ricerca di tutte le qualifiche
Op. 31	Ricerca di tutti i lavoratori

Op. 32	Assegnazione di qualifiche necessarie allo sviluppo di un'applicazione
Op. 33	Inserimento di un lavoratore
Op. 34	Assegnazione di un lavoratore ad un intervento
Op. 35	Modifica/Eliminazione di un lavoratore

2.4 – Vincoli di integrità fondamentali

Impiegati

1. Un impiegato non può lavorare a più di un progetto attivo;
2. Un impiegato non può frequentare corsi che forniscono un insieme di qualifiche che possiede già;
3. Un impiegato non può essere anche collaboratore;
4. Un impiegato può avere esclusivamente qualifiche distinte;

Collaboratori

1. Un collaboratore partecipa ai corsi solo come docente, pertanto non può aumentare le sue qualifiche all'interno della società;
2. Un collaboratore non può essere anche impiegato;
3. Il docente di un corso deve essere unico;

Progetti

1. Non è possibile aggiungere revisioni a un progetto ancora in sviluppo;

Corsi

1. I corsi possono essere frequentati solamente da impiegati che non lavorino nel contempo ad alcun progetto attivo;
2. I corsi possono essere tenuti solamente da collaboratori;

Lavoratori

1. Un lavoratore o è impiegato o è collaboratore;

III – PROGETTAZIONE

3.1 Schema Entity-Relationship Concettuale

3.2 Tavola della frequenza delle operazioni

NOME	OPERAZIONE	FREQUENZA
Op. 1	Inserimento di un nuovo progetto	10/mese
Op. 2	Modifica di un progetto	30/mese
Op. 3	Eliminazione di un progetto	0/mese
Op. 4	Inserimento delle applicazioni software legate al progetto	100/mese
Op. 5	Ricerca dei progetti che hanno superato il periodo di consegna previsto	10/mese
Op. 6	Elenco dei linguaggi utilizzati nello sviluppo di un progetto	10/mese
Op. 7	Elenco delle qualifiche necessarie allo sviluppo di un progetto	30/mese
Op. 8	Verifica della presenza, tra il personale assegnato ad un progetto, di tutte le qualifiche necessarie (associate alle applicazioni contenute)	150/mese
Op. 9	Ricerca dei progetti correntemente in sviluppo e del loro stato di avanzamento (numero delle applicazioni completate)	150/mese
Op. 10	Inserimento di una nuova applicazione	**500/mese**
Op. 11	Modifica dei dati di un'applicazione	**800/mese**
Op. 12	Assegnazione di personale allo sviluppo di un'applicazione	**1000/mese**
Op. 13	Verifica della presenza, tra il personale assegnato ad un'applicazione, di tutte le qualifiche necessarie al completamento dello sviluppo	150/mese
Op. 14	Registrazione degli interventi	**1000/mese**
Op. 15	Inserimento di un nuovo impiegato	1/mese
Op. 16	Modifica/Eliminazione di un impiegato	1/mese
Op. 17	Inserimento dei dati riguardanti i collaboratori (periodo, progetto, compenso)	10/mese
Op. 18	Ricerca di tutto il personale con particolari qualifiche	**500/mese**
Op. 19	Determinazione di tutti i corsi di aggiornamento frequentati da un impiegato	100/mese
Op. 20	Ricerca di tutto il personale attualmente libero (non assegnato ad alcuna applicazione attiva)	**1000/mese**
Op. 21	Inserimento di una nuova qualifica	5/mese
Op. 22	Modifica/Eliminazione di una qualifica	1/mese
Op. 23	Inserimento di un nuovo corso	10/mese
Op. 24	Modifica/Eliminazione di un corso	1/mese
Op. 25	Assegnazione ad un corso delle qualifiche da esso derivanti	100/mese
Op. 26	Aggiunta di partecipanti al corso	200/mese
Op. 27	Aggiornamento delle qualifiche degli impiegati in base ai corsi frequentati	200/mese
Op. 28	Ricerca di tutte le qualifiche di un lavoratore	**500/mese**
Op. 29	Quando la data di rilascio di una applicazione è un valore non nullo, aggiornamento in cascata tutte le tuple della relazione impegno (referenziate da applicazione) che abbiano data_fine nulla	100/mese
Op. 30	Ricerca di tutte le qualifiche	5/mese
Op. 31	Ricerca di tutti i lavoratori	10/mese
Op. 32	Assegnazione di qualifiche necessarie allo sviluppo di	300/mese

	un'applicazione	
Op. 33	Inserimento di un lavoratore	5/mese
Op. 34	Assegnazione di un lavoratore ad un intervento	300/mese
Op. 35	Modifica/Eliminazione di un lavoratore	1/mese

NOTA: le operazioni ritenute critiche sono state riportate in grassetto.

3.3 Tavola del Volume dei Dati

CONCETTO	TIPO	VOLUME
Progetto	Entità	250
Qualifica_necessaria	Entità	7500
Applicazione	Entità	2500
Intervento	Entità	75000
Corso	Entità	240
Abilitazione	Relazione	48000
Qualifica	Entità	200
Partecipazione	Relazione	72000
Possesso	Relazione	600
Lavoratore	Entità	300
Impegno	Relazione	210000
Esecuzione	Relazione	150000
Impiegato	Entità	50
Collaboratore	Entità	250

3.4 Analisi delle Prestazioni

Op. 10 – Inserimento di una nuova applicazione

Concetto	Costrutto	Accessi	Tipo
Applicazione	Entità	1	S
Inclusione	Relazione	1	S
Progetto	Entità	1	L

Costo Operazione = 2S + 1L = 5A * 500v/mese = **2500 A/mese**

Op. 11 – Modifica dei dati di un'applicazione

Concetto	Costrutto	Accessi	Tipo
Applicazione	Entità	1	L
Applicazione	Entità	1	S
Inclusione	Relazione	1	S
Progetto	Entità	1	L

Costo Operazione = 2S + 2L = 6A * 500v/mese = **3000 A/mese**

Op. 12 – Assegnazione di personale allo sviluppo di un'applicazione 1000v/m

Concetto	Costrutto	Accessi	Tipo
Qualifica_Necessaria	Entità	1	L
Possesso	Relazione	3	L
Impegno	Relazione	6	S

Costo Operazione = 6S + 4L = 16A * 1000v/mese = **16000 A/mese**

Op. 14 – Registrazione degli interventi

Concetto	Costrutto	Accessi	Tipo
Intervento	Entità	1	S
Implicazione	Relazione	1	S
Applicazione	Entità	1	L

Costo Operazione = 2S + 1L = 5A * 1000v/mese = **5000 A/mese**

Op. 18 – Ricerca di tutto il personale con particolari qualifiche

Concetto	Costrutto	Accessi	Tipo
Qualifica	Entità	1	L
Possesso	Relazione	10	L

Costo Operazione = 0S + 11L = 11A * 500v/mese = **5500 A/mese**

Op. 20 – Ricerca di tutto il personale attualmente libero

Concetto	Costrutto	Accessi	Tipo
Impegno	Entità	50	L

Costo Operazione = 0S + 50L = 50A * 500v/mese = **25000 A/mese**

Op. 28 – Ricerca di tutte le qualifiche di un lavoratore

Concetto	Costrutto	Accessi	Tipo
Lavoratore	Entità	1	L
Possesso	Relazione	10	L

Costo Operazione = 0S + 11 L = 11A * 500v/mese = **5500 A/mese**

3.5 Analisi delle Ridondanze

Eventuali ridondanze sono state eliminate in fase di progettazione e non, come previsto, in fase di ristrutturazione. Questo è stato reso possibile da un'attenta e minuziosa progettazione.

Al contrario, però, è stato ritenuto opportuno inserire alcune informazioni ridondati per rendere più fluida ed agevole la navigazione all'interno della Base Dati a scapito della memoria.

Considerata l'analisi delle prestazioni ed il relativamente basso costo dell'hardware che supporta la base dati, l'operazione di inserimento di attributi ridondanti nella Base Dati è stata ritenuta estremamente conveniente.

Tali attributi sono:

- Partecipazione.docente;
- Possesso.Corso;
- Impegno.Progetto.

3.6 Schema Entity-Relationship Logico

IV - IMPLEMENTAZIONE

4.1 Mappa Relazionale Completa

Entità PROGETTO

Attributi	Tipo	Unico	Non Nullo	Chiave Primaria	Refereces
ID	integer			X	
Nome	Varchar(50)	X	X		
Descrizione	Text				
Data_Inizio	Date		X		
Data_Debugging	Date				
Data_Rilascio	Date				
Data_Prevista	Date				

Commento

L'entità PROGETTO memorizza una lista di tutti i progetti che la società porta avanti e che ha già consegnato.

La data di inizio di un progetto viene di default settata sulla data attuale.

Considerata l'esperienza che la società ha maturato negli anni si conviene che il termine previsto per la consegna del progetto sia di default uguale ad un mese dalla data di inserimento.

Vincoli di integrità

1. la data d'inizio di un progetto deve essere anteriore, o al più uguale, a quella di rilascio del progetto.
2. non possono esistere due progetti con lo stesso nome.
3. la data di debugging deve essere o compresa tra la data di rilascio e la data di inizio, oppure nulla.
4. la data di rilascio del progetto, se non nulla, deve essere posteriore a quella di inizio.

SQL: Definizione Sequenza

```
create sequence progetto_SEQ start 1
```

SQL: Definizione Tabella

```
create table progetto (
id integer primary key default nextval('Progetto_Seq'),
nome varchar(50) unique not null,
descrizione text default null,
data_inizio date not null default now() check
(controlladataminoredi(data_inizio,data_rilascio)),
data_debugging date default null check
(datacompresa(data_inizio,data_debugging,data_rilascio)),
data_rilascio date default null check
(controlladataminoredi(data_inizio,data_rilascio)),
data_prevista date default date_part('year', now()) || '-' ||
date_part('month', now()) + 1 || '-' || date_part('day', now()))
```

Entità APPLICAZIONE

Attributi	Tipo	Unico	Non Nullo	Chiave Primaria	Refereces
ID	Integer			X	
Nome	Varchar(50)	X*	X		
Descrizione	Varchar(180)				
Data_Rilascio	Date				
Linguaggio	Varchar(30)				
Data_Inizio	Date		X		
Data_Debugging	Date				
Data_Prevista	Date				
Ruolo	Varchar(30)	X*	X		
ID_Progetto	Integer		X		Progetto(ID)

Commento

L'entità APPLICAZIONE memorizza una lista di tutte le applicazioni sviluppate, o in sviluppo, dalla società. Ogni applicazione appartiene ad un solo Progetto.

La data di inizio dell'applicazione viene settata di default sulla data attuale.

Vincoli di integrità

1. la data di inizio di un'applicazione deve essere posteriore, o al più uguale, a quella di inizio del progetto.
2. la data di rilascio di un'applicazione, se non nulla, deve essere anteriore, o al più uguale, a quella di rilascio del progetto.
3. un progetto non può includere più di un'applicazione che svolge lo stesso ruolo.

4. la data di debugging di un'applicazione deve essere, se non nulla, compresa tra la data di inizio e la data di rilascio.
5. la data di rilascio di un'applicazione deve essere, se non nulla, posteriore, o al più uguale, alle date di inizio del progetto e dell'applicazione.
6. Quando alla data di rilascio viene assegnato un valore non nullo devono essere aggiornate in cascata tutte le tuple della relazione Impegno (che ovviamente fanno riferimento all'applicazione) che abbiano data_fine nulla o posteriore alla data di rilascio dell'applicazione.
7. il nome di un'applicazione deve essere unico nell'ambito di ciascun progetto.

Chiavi Esterne

L'attributo ID_Progetto è una chiave esterna che fa riferimento all'ID del Progetto. In situazioni critiche il DBMS deve comportarsi:

- In caso di cancellazione di un progetto: non permettere la cancellazione se esiste almeno un'applicazione che fa riferimento al progetto;
- In caso di aggiornamento di un progetto: aggiorna in cascata tutte le applicazioni che fanno riferimento al progetto.

SQL: Definizione Sequenza

```
create sequence applicazione_SEQ start 1
```

SQL: Dichiarazione Funzioni Specifiche

```
create function ApplicazioneV1(date, integer)
returns boolean as
'declare
data1 alias for $1;
progetto1 alias for $2;
begin
if data1 > (select progetto.data_inizio from progetto where progetto.id =
progetto1) or data1 = (select progetto.data_inizio from progetto where
progetto.id = progetto1)
then
        return "t";
else
  return "f";
end if;
end;'
language 'plpgsql'
```

```
create function ApplicazioneV2(date, integer)
returns boolean as
'declare
```

```
data1 alias for $1;
progetto1 alias for $2;
begin
if data1 < (select data_rilascio from progetto where id = progetto1) or
data1 = (select data_rilascio from progetto where id = progetto1)
then
        return "t";
else
if data1 = null or (select data_rilascio from progetto where id = progetto1)
= null
then
return "t";
else
return "f";
end if;
end if;
end;'
language 'plpgsql'
```

```
create function ApplicazioneV3(date, integer)
returns boolean as
'declare
data1 alias for $1;
applicazione1 alias for $2;
begin
if data1 is not null
then
update qualifica_necessaria, impegno set data_fine = data1 where
id_applicazione = applicazione1 and id_qualifica_necessaria = id and
data_fine is null or data_fine > data1;
return "t";
else
return "t";
end if;
end;'
language 'plpgsql'
```

SQL: Definizione Tabella

```
create table applicazione (
id integer primary key default nextval('Applicazione_Seq'),
nome varchar(50) not null,
descrizione varchar(180),
```

```
data_rilascio date default null check
(ApplicazioneV2(data_rilascio,id_progetto) and
controlladataminoredi(data_inizio,data_rilascio)),
linguaggio varchar(30),
data_inizio date not null default now() check
(ApplicazioneV1(data_inizio,id_progetto) and
controlladataminoredi(data_inizio,data_rilascio)),
data_debugging date default null check
(datacompresa(data_inizio,data_debugging,data_rilascio)),
data_prevista date default null,
ruolo varchar(50) not null,
id_progetto integer not null,
unique (id_progetto,nome),
unique (id_progetto, ruolo),
foreign key (id_progetto) references progetto(id) on delete no action
on update cascade)
```

Entità QUALIFICA_NECESSARIA

Attributi	Tipo	Unico	Non Nullo	Chiave Primaria	Refereces
ID	Integer			X	
Quantita	Integer		X		
ID_Applicazione	Integer	X*	X		Applicazione(ID)
ID_Qualifica	Varchar(30)	X*	X		Qualifica(Nome)

Commento

L'entità Qualifica_necessaria memorizza una lista di tutte le qualifiche e le rispettive quantità che necessitano per lo sviluppo di tutte le applicazioni che la società ha sviluppato o che deve sviluppare.

Di default il campo Quantità viene settato sul valore 1.

Vincoli di integrità

1. la quantità di lavoratori con determinate qualifiche richiesti per sviluppare un'applicazione non può essere minore di 1.
2. la quantità di lavoratori con determinate qualifiche richieste per sviluppare un'applicazione non può essere maggiore del numero di tutti i lavoratori dell'azienda.

3. un'applicazione non può avere due identiche qualifiche necessarie, anche se la quantità di lavoratori richiesta è diversa. (unique(id_applicazione, id_qualifica).
4. non possono esistere più di Qualifica_Necessaria(quantita) lavoratori che impegnano una qualifica necessaria ad un'applicazione.

Chiavi Esterne

L'attributo ID_Applicazione è una chiave esterna che fa riferimento all'ID dell'Applicazione. In situazioni critiche il DBMS deve comportarsi:

- in caso di cancellazione di un'applicazione: cancella in cascata tutte le tuple che vi fanno riferimento;
- in caso di aggiornamento di un'applicazione: aggiorna in cascata tutte le tuple che vi fanno riferimento;

L'attributo ID_Qualifica è una chiave esterna che fa riferimento al NOME della Qualifica. In situazioni critiche il DBMS deve comportarsi:

- in caso di cancellazione di una qualifica: non permettere la cancellazione se esiste almeno una tupla che vi fa riferimento;
- in caso di aggiornamento di un'applicazione: aggiorna in cascata tutte le tuple che vi fanno riferimento.

SQL: Definizione Sequenza

```
create sequence Qualifica_Necessaria_SEQ start 1
```

SQL: Dichiarazione Funzioni Specifiche

```
create function qualifica_necessariaV1() returns integer as
'select count(*) + 1 as result from lavoratore'
language 'sql'
```

SQL: Definizione Tabella

```
create table qualifica_necessaria (
id integer primary key default nextval('qualifica_necessaria_Seq'),
quantita integer not null default 1 check (quantita <
qualifica_necessariaV1() and quantita > 0),
ID_Applicazione integer not null,
ID_qualifica varchar(30)not null,
unique (ID_Applicazione,ID_Qualifica),
foreign key (ID_Applicazione) references applicazione(ID) on delete
cascade on update cascade,
```

```
foreign key (ID_qualifica) references qualifica(nome) on delete no action on update cascade
)
```

Entità INTERVENTO

Attributi	Tipo	Unico	Non Nullo	Chiave Primaria	Refereces
ID	Integer			X	
ID_Applicazione	Integer		X		Applicazione(ID)
Descrizione	Varchar(180)				
Tipo	Integer		X		
Data	Date		X		

Commento

L'entità Intervento memorizza una lista di tutti gli interventi di correzione, aggiornamento e modifica di tutte le applicazioni sviluppate e in sviluppo dalla società.

Il tipo di intervento viene identificato con un codice intero che assume uno dei seguenti significati:

0 = Modifica
1 = Correzione
2 = Aggiornamento

Di default il valore assegnato a Tipo è 0.

Di default la data di inizio viene settata sulla data attuale.

Vincoli di integrità

1. Un intervento non può essere eseguito se l'applicazione non è stata ancora rilasciata.
2. Tipo deve essere compreso tra 0 e 2.

Chiavi Esterne

L'attributo ID_Applicazione è una chiave esterna che fa riferimento all'ID dell'Applicazione. In situazioni critiche il DBMS deve comportarsi:

- in caso di cancellazione di un'applicazione: cancella in cascata tutte le tuple che vi fanno riferimento;
- in caso di aggiornamento di un'applicazione: aggiorna in cascata tutte le tuple che vi fanno riferimento.

SQL: Definizione Sequenza

```
create sequence intervento_SEQ start 1
```

SQL: Dichiarazione Funzioni Specifiche

```
create function interventoV1(integer)
returns boolean as
'declare
applicazioneRIF alias for $1;
rec record;
begin
select into rec * from applicazione where id = applicazioneRIF and
data_rilascio = null;
if found
then
        return "f";
else
  return "t";
end if;
end;'
language 'plpgsql'
```

SQL: Definizione Tabella

```
create table intervento (
id integer primary key default nextval('intervento_seq'),
id_applicazione integer not null check (interventoV1(id_applicazione)),
descrizione varchar(180) default null,
tipo integer not null default 0 check (tipo between 0 and 2),
data date not null default now(),
foreign key (id_applicazione) references applicazione(id) on delete
cascade on update cascade
)
```

Relazione IMPEGNO

Attributi	Tipo	Unico	Non Nullo	Chiave Primaria	Refereces
ID_Qualifica_Necessaria	Integer			X*	Qualifica_Necessaria(ID)
ID_Lavoratore	Varchar(16)			X*	Lavoratore(Cod_Fiscale)

Data_inizio	Date		X		
Data_fine	Date				
Compenso	Decimal(4,2)				
Progetto	Integer				Progetto(id)

Commento

La relazione Impegno memorizza una lista di tutti i periodi e delle qualifiche con un lavoratore ha partecipato allo sviluppo di un'applicazione.

Il campo compenso è settato di default sul valore nullo, in quanto ha senso esclusivamente per un collaboratore.

La data di inizio, di default, viene settata sulla data attuale.

L'attributo progetto è stato aggiunto nella fase di ottimizzazione per consentire una navigazione nel database ad un costo estremamente minore.

Esso di default assume il valore della chiave primaria del progetto a cui si riferisce.

Vincoli di integrità

1. non possono esistere più di Qualifica_Necessaria(Quantita) lavoratori che impegnano una qualifica necessaria ad un'applicazione.
2. la data di fine deve essere o nulla o posteriore, o al più uguale, alla data di inizio.
3. la data di fine deve essere anteriore, o al più uguale, alla data di rilascio dell'applicazione.
4. un impiegato non può partecipare a più di un progetto attivo.
5. il compenso, se non nullo, deve essere maggiore di 0.

Chiavi Esterne

L'attributo ID_Qualifica_Necessaria è una chiave esterna che fa riferimento all'ID della Qualifica_Necessaria. In situazioni critiche il DBMS deve comportarsi:

- in caso di cancellazione di una qualifica necessaria: cancella in cascata tutte le tuple che vi fanno riferimento;
- in caso di aggiornamento di una qualifica necessaria: aggiorna in cascata tutte le tuple che vi fanno riferimento.

L'attributo ID_Lavoratore è una chiave esterna che fa riferimento al codice fiscale di Lavoratore. In situazioni critiche il DBMS deve comportarsi:

- in caso di cancellazione di un lavoratore: non consentire l'operazione se esiste almeno una tupla che vi fa riferimento;
- in caso di aggiornamento di un lavoratore: aggiorna in cascata tutte le tuple che vi fanno riferimento.

L'attributo Progetto è una chiave esterna che fa riferimento all'ID del progetto. In situazioni critiche il DBMS deve comportarsi:

- in caso di cancellazione di un progetto: cancella in cascata tutte le tuple che vi fanno riferimento;
- in caso di aggiornamento di un progetto: aggiorna in cascata tutte le tuple che vi fanno riferimento.

SQL: Dichiarazione Funzioni Specifiche

```
create function impegnoV1(date, integer)
returns boolean as
'declare
data1 alias for $1;
applicazione1 alias for $2;
begin
if data1 < (select applicazione.data_rilascio  from applicazione,
qualifica_necessaria where qualifica_necessaria.id = applicazione1 and
qualifica_necessaria.id_applicazione = applicazione.id) or data1 =
(select applicazione.data_rilascio  from applicazione,
qualifica_necessaria where qualifica_necessaria.id = applicazione1 and
qualifica_necessaria.id_applicazione = applicazione.id)
or data1 = null or (select applicazione.data_rilascio  from
applicazione, qualifica_necessaria where qualifica_necessaria.id =
applicazione1 and qualifica_necessaria.id_applicazione =
applicazione.id) = null
then
       return "t";
else
  return "f";
end if;
end;'
language 'plpgsql'
```

```
create function impegnoV2(integer) returns integer as
'select quantita - (select count(*) from impegno where
id_qualifica_necessaria = $1) as result from  qualifica_necessaria
where qualifica_necessaria.id = $1'
language 'sql'
```

```
create function impegnoV3(varchar(16))
returns boolean as
'declare
codfisc alias for $1;
lavoratoreRIF record;
begin
select into lavoratoreRIF * from  impegno where id_lavoratore =
```

```
codfisc and data_fine is null or data_fine > now();
if found
then
  return "f";
else
  return "t";
end if;
end;'
language 'plpgsql'
```

```
create function impegnoV4(integer) returns integer as
'select progetto.id as result from progetto, applicazione,
qualifica_necessaria where progetto.id = id_progetto and
applicazione.id = id_applicazione and qualifica_necessaria.id = $1'
language 'sql'
```

```
create function impegnoV5(varchar(16),integer)
returns boolean as
'declare
codfisc alias for $1;
qual alias for $2;
lavoratoreRIF record;
begin
select into lavoratoreRIF * from  possesso, qualifica_necessaria where
qualifica_necessaria.id = qual and id_lavoratore = codfisc and
qualifica_necessaria.id_qualifica = possesso.id_qualifica ;
if found
then
  return "t";
else
  return "f";
end if;
end;'
language 'plpgsql'
```

```
create function ImpegnoV6(integer) returns integer as
'select count(*) from impegno where id_qualifica_necessaria = $1
group by id_qualifica_necessaria'
language 'sql'
```

SQL: Definizione Tabella

```
create table impegno (
id_qualifica_necessaria integer,
id_lavoratore varchar(16) check (impegnoV3(id_lavoratore) and
impegnoV5(id_lavoratore,id_qualifica_necessaria)),
data_inizio date not null check (controlladataminoredi(data_inizio,
data_fine)),
data_fine date check (impegnoV1(data_fine,id_qualifica_necessaria)),
compenso decimal(4,2) default null check (compenso > 0 or
compenso is null),
progetto integer check (progetto =
impegnoV4(id_qualifica_necessaria)),
primary key (id_qualifica_necessaria, id_lavoratore),
foreign key (id_qualifica_necessaria) references
qualifica_necessaria(id) on delete cascade on update cascade,
foreign key (id_lavoratore) references lavoratore(cod_fiscale) on delete
no action on update cascade,
foreign key (progetto) references progetto(id) on delete cascade on
update cascade
)
```

Relazione ESECUZIONE

Attributi	Tipo	Unico	Non Nullo	Chiave Primaria	Refereces
ID_Intervento	Integer			X*	Intervento(ID)
ID_Lavoratore	Varchar(16)			X*	Lavoratore(Cod_Fiscale)

Commento

La relazione Esecuzione memorizza una lista di tutti i gli interventi eseguiti da ciascun lavoratore.

Vincoli di integrità

1. deve essere unica la coppia (id_intervento, id_lavoratore).

Chiavi Esterne

L'attributo ID_Intervento è una chiave esterna che fa riferimento all'ID dell'Intervento. In situazioni critiche il DBMS deve comportarsi:

- in caso di cancellazione di un intervento: cancella in cascata tutte le tuple che vi fanno riferimento;
- in caso di aggiornamento di una qualifica necessaria: aggiorna in cascata tutte le tuple che vi fanno riferimento.

L'attributo ID_Lavoratore è una chiave esterna che fa riferimento al codice fiscale di Lavoratore. In situazioni critiche il DBMS deve comportarsi:

- in caso di cancellazione di un lavoratore: non consentire l'operazione se esiste almeno una tupla che vi fa riferimento;
- in caso di aggiornamento di un lavoratore: aggiorna in cascata tutte le tuple che vi fanno riferimento.

SQL: Definizione Tabella

```
create table esecuzione (
id_intervento integer,
id_lavoratore varchar(16),
primary key (id_intervento, id_lavoratore),
foreign key (id_intervento) references intervento(id) on delete cascade
on update cascade,
foreign key (id_lavoratore) references lavoratore(cod_fiscale)  on
delete no action on update cascade
)
```

Entità LAVORATORE

Attributi	Tipo	Unico	Non Nullo	Chiave Primaria	Refereces
Cod_Fiscale	Varchar(16)			X	
Cognome	Varchar(25)		X		
Nome	Varchar(30)		X		

Commento

L'entità Lavoratore memorizza una lista di tutti i lavoratori della società.

Vincoli di integrità

1. non possono coesistere due lavoratori con lo stesso codice fiscale.

SQL: Definizione Sequenza

Poiché la chiave primaria è il Codice Fiscale (di tipo varchar), non esistono sequenze per questa entità.

SQL: Definizione Tabella

```
create table Lavoratore (
cod_fiscale varchar(16) primary key,
Cognome varchar(25) not null,
Nome varchar(30) not null
)
```

Entità IMPIEGATO

Attributi	Tipo	Unico	Non Nullo	Chiave Primaria	Refereces
Cod_Fiscale	Varchar(16)			X	Lavoratore(Cod_Fiscale)
Data_Assunzione	Date		X		
Stipendio	Decimal(6,2)		X		
Matricola	Integer	X	X		

Commento

L'entità Impiegato memorizza una lista di tutti i lavoratori impiegati nella società.
Si conviene per semplicità che la matricola di un impiegato sia di tipo numerico.

Vincoli di integrità

1. non possono coesistere due impiegati con lo stesso codice fiscale.
2. non possono coesistere due impiegati con la stessa matricola.
3. lo stipendio deve essere maggiore di 0.

Chiavi Esterne

L'attributo Cod_Fiscale è una chiave esterna che fa riferimento al codice fiscale del Lavoratore. In situazioni critiche il DBMS deve comportarsi:

- in caso di cancellazione di un lavoratore: cancella in cascata tutte le tuple che vi fanno riferimento;
- in caso di aggiornamento di un lavoratore: aggiorna in cascata tutte le tuple che vi fanno riferimento.

SQL: Definizione Sequenza

Poiché la chiave primaria è il Codice Fiscale (di tipo varchar), non esistono sequenze per questa entità.

SQL: Definizione Tabella

```
create table impiegato (
cod_fiscale varchar(16) primary key,
data_assunzione date not null,
stipendio decimal(6,2) not null default 0 check (stipendio > 0),
matricola integer unique not null,
foreign key (cod_fiscale) references lavoratore(cod_fiscale) on delete
cascade on update cascade
)
```

Relazione PARTECIPAZIONE

Attributi	Tipo	Unico	Non Nullo	Chiave Primaria	Refereces
ID_Corso	Integer			X*	Corso(ID)
ID_Lavoratore	Varchar(16)			X*	Lavoratore(Cod_Fiscale)
Docente	Boolean		X		

Commento

La relazione Pertecipazione memorizza una lista di tutti i lavoratori che hanno partecipato ai corsi organizzati dalla società.

L'attributo docente è stato inserito nella fase di ottimizzazione per consentire una navigazione nel database con costi minori.

L'attributo docente è vero solo se il lavoratore non è un impiegato.

Di default il valore viene settato su falso.

Vincoli di integrità

1. deve essere unica la coppia (id_lavoratore, id_corso).
2. un impiegato che partecipa ad un corso non vi partecipa come docente.
3. un collaboratore che partecipa ad un corso vi partecipa solo come docente e pertanto non può incrementare le proprie qualifiche.

4. un impiegato non può partecipare ad un corso che fornisce un insieme di qualifiche che possiede già.
5. possono partecipare ai corsi solo gli impiegati che non lavorano, al momento, in alcun progetto attivo

Chiavi Esterne

L'attributo ID_Corso è una chiave esterna che fa riferimento all'ID del Corso. In situazioni critiche il DBMS deve comportarsi:

- in caso di cancellazione di un corso: cancella in cascata tutte le tuple che vi fanno riferimento;
- in caso di aggiornamento di un corso: aggiorna in cascata tutte le tuple che vi fanno riferimento.

L'attributo ID_Lavoratore è una chiave esterna che fa riferimento al codice fiscale di Lavoratore. In situazioni critiche il DBMS deve comportarsi:

- in caso di cancellazione di un lavoratore: non consentire l'operazione se esiste almeno una tupla che vi fa riferimento;
- in caso di aggiornamento di un lavoratore: aggiorna in cascata tutte le tuple che vi fanno riferimento.

SQL: Dichiarazione funzioni specifiche

```
create function partecipazioneV1(varchar(16))
returns boolean as
'declare
lavoratoreRIF alias for $1;
lavor record;
begin
select into lavor *  from lavoratore  where cod_fiscale = lavoratoreRIF
and  cod_fiscale in (select cod_fiscale from impiegato);
if found
then
        return "f";
else
  return "t";
end if;
end;'
language 'plpgsql'
```

```
create function partecipazioneV2(varchar(16), integer)
returns boolean as
'declare
codfisc alias for $1;
```

```
codcorso alias for $2;
cerca record;
cerca2 record;
begin

select into cerca2 cod_fiscale from impiegato where cod_fiscale =
codfisc;

if found
then
select into cerca id_qualifica from abilitazione, corso where id_corso =
corso.id and corso.id = codcorso
except
select id_qualifica from possesso, lavoratore where id_lavoratore =
cod_fiscale and id_lavoratore = codfisc;
if found
then
        return "t";
else
  return "f";
end if;
else
  return "t";
end if;

end;'
language 'plpgsql'
```

SQL: Definizione Tabella

```
create table partecipazione (
id_corso integer check (partecipazioneV2(id_lavoratore, id_corso)),
id_lavoratore varchar(16) check (partecipazioneV2(id_lavoratore,
id_corso) and impegnoV3(id_lavoratore)),
docente boolean not null check (docente =
partecipazioneV1(id_lavoratore)),
primary key (id_corso, id_lavoratore),
foreign key (id_corso) references corso(id) on delete cascade on
update cascade,
foreign key (id_lavoratore) references lavoratore(cod_fiscale) on delete
no action on update cascade
)
```

Entità CORSO

Attributi	Tipo	Unico	Non Nullo	Chiave Primaria	Refereces
ID	Integer			X	
Nome	Varchar(50)	X*	X		
Ore	Integer		X		
Data_Inizio	Date	X*	X		
Data_Fine	Date				

Commento

L'entità Corso memorizza una lista di tutti i corsi che la società organizza ed ha organizzato.
Il campo impiegato è stato aggiunto per ottimizzare il sistema in velocità.
Di default questo campo è falso.
Di default il campo ore è settato su 1.

Vincoli di integrità

1. deve essere unica la coppia (Nome,Data_Inizio).
2. il numero di ore non può essere minore di 1.
3. la data di fine di un corso, se non nulla, deve essere posteriore, o al più uguale, della data di inizio.

SQL: Definizione Sequenza

```
create sequence Corso_SEQ start 1
```

Relazione ABILITAZIONE

Attributi	Tipo	Unico	Non Nullo	Chiave Primaria	Refereces
ID_Corso	Integer			X*	Corso(ID)
ID_Qualifica	Varchar(30)			X*	Qualifica(Nome)

Commento

La relazione Abilitazione memorizza una lista di tutte le qualifiche fornite da ciascun corso organizzato dalla società.

Vincoli di integrità

1. deve essere unica la coppia (ID_Qualifica, ID_Corso).

Chiavi Esterne

L'attributo ID_Corso è una chiave esterna che fa riferimento all'ID del Corso. In situazioni critiche il DBMS deve comportarsi:

- in caso di cancellazione di un corso: cancella in cascata tutte le tuple che vi fanno riferimento;
- in caso di aggiornamento di un corso: aggiorna in cascata tutte le tuple che vi fanno riferimento.

L'attributo ID_Qualifica è una chiave esterna che fa riferimento al nome della Qualifica. In situazioni critiche il DBMS deve comportarsi:

- in caso di cancellazione di una qualifica: non consentire l'operazione se esiste almeno una tupla che vi fa riferimento;
- in caso di aggiornamento di una qualifica: aggiorna in cascata tutte le tuple che vi fanno riferimento.

Entità QUALIFICA

Attributi	Tipo	Unico	Non Nullo	Chiave Primaria	Refereces
Nome	Varchar(30)			X	

Commento

L'entità Qualifica memorizza una lista di tutte le qualifiche conosciute dalla società.

SQL: Definizione Sequenza

Poiché la chiave primaria è il nome (di tipo varchar), non esistono sequenze per questa entità.

Relazione POSSESSO

Attributi	Tipo	Unico	Non	Chiave	Refereces

			Nullo	Primaria	
ID_Lavoratore	Varchar(16)			X*	Lavoratore(Cod_Fiscale)
ID_Qualifica	Varchar(30)			X*	Qualifica(Nome)
Corso	Integer				

Commento

La relazione Possesso memorizza una lista di tutte le qualifiche possedute da ciascun lavoratore.

Il campo corso è stato aggiunto in fase di ottimizzazione per consentire una navigazione nel database con costi minori.

Vincoli di integrità

1. deve essere unica la coppia (ID_Qualifica, ID_Lavoratore).
2. il campo corso deve essere settato uguale all'ID del corso che ha fornito la qualifica all'impiegato.
3. il campo corso è nullo se il lavoratore che possiede questa qualifica è un collaboratore.

Chiavi Esterne

L'attributo ID_Lavoratore è una chiave esterna che fa riferimento al codice fiscale del Lavoratore. In situazioni critiche il DBMS deve comportarsi:

- in caso di cancellazione di un lavoratore: cancella in cascata tutte le tuple che vi fanno riferimento;
- in caso di aggiornamento di un lavoratore: aggiorna in cascata tutte le tuple che vi fanno riferimento.

L'attributo ID_Qualifica è una chiave esterna che fa riferimento al nome della Qualifica. In situazioni critiche il DBMS deve comportarsi:

- in caso di cancellazione di una qualifica: non consentire l'operazione se esiste almeno una tupla che vi fa riferimento;
- in caso di aggiornamento di una qualifica: aggiorna in cascata tutte le tuple che vi fanno riferimento.

4.2 Dichiarazione Funzioni Globali e Triggers

Datacompresa(date, date, date)

```
create function datacompresa(date, date, date)
returns boolean as
'declare
data1 alias for $1;
data2 alias for $2;
data3 alias for $3;
begin
if (data1 < data2 or data1 = data2) and (data2 < data3 or data2 =
data3)
then
       return "t";
else
       if data1 = null or data2 = null or data3 = null
  then
     return "t";
  else
     return "f";
  end if;
end if;
end;'
language 'plpgsql'
```

Controlladataminoredi(date, date)

```
create function controlladataminoredi(date, date)
returns boolean as
'declare
data1 alias for $1;
data2 alias for $2;
begin
if data1 < data2 or data1 = data2
then
       return "t";
else
       if data1 = null or data2 = null
  then
     return "t";
```

```
  else
    return "f";
  end if;
end if;
end;'
language 'plpgsql'
```

4.3 Traduzione delle Operazioni previste sulla Base Dati

Op. 1 – Inserimento di un nuovo progetto

```
insert into lavoratore (cod_fiscale, nome, cognome) values (
§!codfiscale§'§Inserisci il codice fiscale:!§,

§!nome§'§Inserisci il Nome:!§,

§!cognome§'§Inserisci il Cognome:!§
)
```

Op. 2– Modifica di un progetto

```
update progetto set §.data_rilascio§'§Imposta la nuova data di rilascio
per il progetto:.§ where id = §!id§§Seleziona il progetto che intendi
modificare:§select id, nome from progetto order by nome!§
```

Op. 3 – Eliminazione di un progetto

```
delete from progetto where id = §!nome_progetto§§Scegli il progetto
che intendi eliminare:§select id, nome from progetto order by nome!§
```

Op. 4 – Inserimento delle applicazioni legate al progetto

```
insert into applicazione (nome, ruolo,descrizione, linguaggio, id_progetto)
values
(§!nome§'§Inserisci il nome:!§,
§!ruolo§'§Inserisci il ruolo:!§,
```

```
§!descrizione§'§Inserisci una breve descrizione:!§,
§!linguaggio§'§Specifica il linguaggio con cui è stata sviluppata questa applicazione:!§,
§!id_progetto§§Scegli il progetto di riferimento:§select id, nome from progetto order by nome!§)
```

Op. 5 – Ricerca dei progetti che hanno superato il periodo di consegna previsto

```
select nome from progetto where data_rilascio > data_prevista or (data_rilascio is null and data_prevista < now())
```

Op. 6 – Elenco dei linguaggi utilizzati nello sviluppo di un progetto

```
select distinct linguaggio from applicazione, progetto where id_progetto = §!id_progetto§§Inserisci il ruolo:§select id, nome from progetto order by nome!§
```

Op. 7 – Elenco delle qualifiche necessarie allo sviluppo di un progetto

```
select id_qualifica, sum(quantita) as quantita_richiesta from qualifica_necessaria, applicazione, progetto where progetto.id = §!progetto.id§§Seleziona il progetto di riferimento:§select id, nome from progetto order by nome!§ and applicazione.id_progetto = progetto.id and qualifica_necessaria.id_applicazione = applicazione.id group by id_qualifica
```

Op. 8 – Verifica della presenza, tra il personale assegnato ad un progetto, di tutte le qualifiche necessarie

```
select id_qualifica,quantita, (select count(*) from impegno where id_qualifica_necessaria = qualifica_necessaria.id) as assegnati from qualifica_necessaria, applicazione where applicazione.id = id_applicazione and id_progetto = §!id_progetto§§Scegli il progetto di riferimento:§select id,nome from progetto order by nome!§
```

Op. 9 – Ricerca dei progetti correntemente in sviluppo e del loro stato di avanzamento

```
select progetto.nome, 'realizzate ' ||
(select count(*) from progetto, applicazione where progetto.data_rilascio is
not null and applicazione.id_progetto = progetto.id) || ' applicazioni su ' ||
(select count(*) from progetto, applicazione where applicazione.id_progetto
= progetto.id) as stato
```

Op. 10 – Inserimento di una nuova applicazione

Accorpata all'operazione 4

Op. 11 – Modifica dei dati di un'applicazione

```
update applicazione set §.data_rilascio§'§Imposta la nuova data di
rilascio per l'applicazione:.§ where id = §!id§§Seleziona l'applicazione
che intendi modificare:§select id, nome from applicazione order by
nome!§
```

Op. 12 – Assegnazione di Personale allo sviluppo di un'applicazione

```
insert into impegno (progetto,id_qualifica_necessaria, id_lavoratore,
data_inizio, data_fine, compenso) values (
§!progetto§§Scegli il progetto di appartenza:§select id, nome from
progetto order by nome!§,

§!id_qualifica_necessaria§§Scegli il lavoro che vuoi assegnare:§
select qualifica_necessaria.id, id_qualifica, applicazione.nome from
qualifica_necessaria, applicazione where applicazione.id =
id_applicazione and applicazione.data_rilascio is null and (quantita <
impegnoV6(qualifica_necessaria.id) or qualifica_necessaria.id not in
(select distinct id_qualifica_necessaria from impegno))!§,

§!id_lavoratore§'§Il sistema ha estratto tutti i lavoratori che non sono
al momento impegnati in alcun progetto attivo.Scegline uno:§
select cod_fiscale, nome, cognome from lavoratore where
impegnoV3(cod_fiscale) = true!§,

§!data_inizio§'§Data inizio:!§,

§!data_fine§'§Data fine:!§,

§!compenso§'§Compenso:!§
)
```

Op. 13 – Verifica della presenza, tra il personale assegnato ad un'applicazione, di tutte le qualifiche necessarie al completamento dello sviluppo

```
select id_qualifica,quantita, (select count(*) from impegno where
id_qualifica_necessaria = id) as assegnati from qualifica_necessaria
where id_applicazione = §!id_applicazione§§Scegli l'applicazione di
riferimento:§select id,nome from applicazione order by nome!§
```

Op. 14 – Registrazione degli interventi

```
insert into intervento (descrizione, tipo, id_applicazione) values
(§!descrizione§'§Inserisci la descrizione dell'intervento:!§,

§!tipo§§Specifica il tipo di intervento:
1 = correzione
2 = aggiornamento!§,

§!id_applicazione§§Scegli l'applicazione di riferimento:§select id,
nome from applicazione where data_rilascio is not null order by
nome!§)
```

Op. 15 – Inserimento di un nuovo impiegato

```
insert into impiegato (cod_fiscale,matricola, data_assunzione,
stipendio) values (
§!codfiscale§'§Seleziona il lavoratore che intendi trasformare in
impiegato:§select cod_fiscale, cognome, nome from lavoratore where
cod_fiscale not in (select cod_fiscale from impiegato) order by
cognome, nome!§,
§!matricola§§Inserisci la matricola:!§,
§!data_assunzione§'§Inserisci la data di assunzione:!§,
§!stipendio§'§Inserisci lo stipendio attuale:!§)
```

Op. 16A – Modifica dei dati di un impiegato

```
update impiegato set §.stipendio§§Imposta il nuovo stipendio
dell'impiegato:.§ where cod_fiscale = §!cod_fiscale §'§Seleziona
l'impiegato di cui intendi modificare lo stipendio:§select cod_fiscale,
cognome, nome from lavoratore where cod_fiscale in (select
cod_fiscale from impiegato) order by cognome, nome!§
```

Op. 16B – Eliminazione di un impiegato

```
delete from impiegato where cod_fiscale = §!cod_fiscale §'§Seleziona l'impiegato che vuoi rimuovere dall'archivio:§select cod_fiscale, cognome, nome from lavoratore where cod_fiscale in (select cod_fiscale from impiegato) order by cognome, nome!§
```

Op. 17 – Inserimento dei dati riguardanti i collaboratori

Questa operazione è stata in parte accorpata pienamente all'operazione 12

Op. 18 – Ricerca di tutto il personale con particolari qualifiche

```
select cognome, lavoratore.nome from lavoratore, possesso where cod_fiscale = id_lavoratore and id_qualifica = §!id_qualifica§'§Seleziona una qualifica:§select nome from qualifica order by nome!§
```

Op. 19 – Determinazione di tutti i corsi di aggiornamento frequentati da un impiegato

```
select nome, data_inizio, data_fine, ore from corso, partecipazione where id_corso = corso.id and id_lavoratore = §!id_lavoratore§'§Seleziona il lavoratore di cui vuoi conoscere i corsi che ha frequentato:§select cod_fiscale, cognome, nome from lavoratore where cod_fiscale in (select cod_fiscale from impiegato) order by cognome, nome!§
```

Op. 20 – Ricerca di tutto il personale attualmente libero (non assegnato ad alcuna applicazione attiva)

```
select nome, cognome from lavoratore where impegnoV3(cod_fiscale) = true
```

Op. 21 – Inserimento di una nuova qualifica

```
insert into qualifica (nome) values
(§!nome§'§Inserisci il nome della qualifica che vuoi inserire:!§)
```

Op. 22A – Modifica di una qualifica

```
insert into qualifica (nome) values
(§!nome§'§Inserisci il nome della qualifica che vuoi inserire:!§)
```

Op. 22B – Eliminazione di una qualifica

```
delete from qualifica where (§.nome§'§Seleziona la qualifica da
rimuovere:§select nome from qualifica order by nome.§)
```

Op. 23 – Inserimento di un nuovo corso

```
insert into corso (nome, ore, data_inizio, data_fine) values (
§!nome§'§Inserisci il nome del corso:!§,
§!ore§§Inserisci il numero di ore:!§,
§!data_inizio§'§Inserisci la data in cui il corso ha inizio:!§,
§!data_fine§'§Inserisci la data in cui il corso avrà fine:!§
)
```

Op. 24A – Modifica di un corso

```
update corso set data_fine = §!data_fine§'§Inserisci la nuova data in
cui il corso terminerà:!§, ore = §!ore§§Inserisci il nuovo numero di
ore:!§ where id = §!id§§Seleziona il corso da modificare:§select id,
nome, data_inizio from corso order by data_inizio, nome!§
```

Op. 24B – Eliminazione di un corso

```
delete from corso where id =
§!id§§Seleziona il corso da modificare:§select id, nome, data_inizio
from corso order by data_inizio, nome!§
```

Op. 25 – Assegnazione ad un corso delle qualifiche da esso derivanti

```
insert into abilitazione (id_corso, id_qualifica) values (
§!id_corso§§Seleziona il corso a cui vuoi assegnare delle qualifiche
derivanti:§select id, nome, data_inizio from corso order by
data_inizio, nome!§,
§!id_qualifica§'§Seleziona una qualifica:§select nome from qualifica
order by nome!§)
```

Op. 26A – Aggiunta di partecipanti ad un corso

```
insert into partecipazione (id_corso, id_lavoratore, docente) values (
§!id_corso§§Seleziona un corso:§select id, nome, data_inizio from
corso where  data_fine is null or data_fine < now() order by
data_inizio, nome!§,
§!id_lavoratore§'§Seleziona l'impiegato che vuoi far partecipare al
corso:§select cod_fiscale, cognome, nome from lavoratore where
cod_fiscale in (select cod_fiscale from impiegato) order by cognome,
nome!§,
false)
```

Op. 26B – Aggiunta di docenti ad un corso

```
insert into partecipazione (id_corso, id_lavoratore, docente) values (
§!id_corso§§Seleziona un corso:§select id, nome, data_inizio from
corso where  data_fine is null or data_fine < now() order by
data_inizio, nome!§,
§!id_lavoratore§'§Seleziona il collaboratore che terrà il corso di
aggiornamento specificato:§select cod_fiscale, cognome, nome from
lavoratore where cod_fiscale not in (select cod_fiscale from impiegato)
order by cognome, nome!§,
true)
```

Op. 27 – Aggiornamento delle qualifiche degli impiegati in base ai corsi frequentati

```
insert into possesso  (
select  partecipazione.id_lavoratore,
abilitazione.id_qualifica,
abilitazione.id_corso
from abilitazione, partecipazione
where
docente = false and
partecipazione.id_corso = §!id_corso§§Seleziona un corso:§select id,
nome, data_inizio from corso order by data_inizio, nome!§ )
```

Op. 28 – Ricerca di tutte le qualifiche di un lavoratore

```
select id_qualifica from possesso where id_lavoratore =
§!id_lavoratore§'§Seleziona un lavoratore:§select cod_fiscale,
cognome, nome from lavoratore order by cognome, nome!§
```

Op. 29 – Ricerca di tutte le applicazioni di un progetto

```
select * from progetto, applicazione where applicazione.id_progetto =
progetto.id and progetto.id = §!id_progetto§§Scegli il progetto di cui
vuoi conoscere le applicazioni:§select id, nome from progetto order
by nome!§
```

Op. 30 – Ricerca di tutte le qualifiche

```
select * from qualifica order by nome
```

Op. 31 – Ricerca di tutti i lavoratori

```
select nome, cognome, cod_fiscale from lavoratore
```

Op. 32 – Assegnazione di qualifiche necessarie allo sviluppo di un'applicazione

```
insert into qualifica_necessaria (id_applicazione, id_qualifica, quantita)
values (§!id_applicazione§§Scegli l'applicazione a cui ti
riferisci:§select applicazione.id, applicazione .nome, progetto.nome
from applicazione, progetto order by progetto.nome,
applicazione.nome!§,

§!id_qualifica§'§Scegli la qualifica da inserire:§select nome from
qualifica order by nome!§,

§!quantita§'§Specifica la quantità:!§)
```

Op. 33 – Inserimento di un lavoratore

```
insert into lavoratore (cod_fiscale, nome, cognome) values (
§!codfiscale§'§Inserisci il codice fiscale:!§,

§!nome§'§Inserisci il Nome:!§,

§!cognome§'§Inserisci il Cognome:!§)
```

Op. 34 – Assegnazione di un lavoratore ad un intervento

```
insert into esecuzione (id_lavoratore, id_intervento) values (

§!id_lavoratore§'§Il sistema ha estratto tutti i lavoratori che non sono
al momento impegnati in alcun progetto attivo. Scegline uno:§
select cod_fiscale, nome, cognome from lavoratore where
impegnoV3(cod_fiscale) = true!§,

§!id_intervento§§Seleziona un intervento:§
select id,descrizione from intervento except select id_intervento,
descrizione from intervento, esecuzione where id_intervento =
intervento.id!§)
```

Op. 35A – Modifica dei dati riguardanti un lavoratore

```
update lavoratore  set nome = §!nome§'§Inserisci il nuovo nome:!§,
cognome = §!cognome§'§Inserisci il nuovo cognome:!§,
cod_fiscale = §!cod_fiscale§'§Inserisci il nuovo codice fiscale:!§
where cod_fiscale =
§!cod_fiscale§'§Seleziona il lavoratore che vuoi modificare:§select
cod_fiscale, cognome, nome from lavoratore order by cognome,
nome!§
```

Op. 35B – Eliminazione di un lavoratore

```
delete from lavoratore  where cod_fiscale =
§!cod_fiscale§'§Seleziona il lavoratore che vuoi rimuovere:§select
cod_fiscale, cognome, nome from lavoratore order by cognome,
nome!§
```

V - INTERFACCIA: EZquery

5.1 Documentazione

Ezquery è un software realizzato con lo scopo di fornire un'interfaccia di basso livello (con istruzioni direttamente in codice SQL) per manipolare databases di PostgreSQL.

Ezquery è stato appositamente realizzato da questo gruppo come complemento al presente progetto.

Principali caratteristiche:

- lettura semplificata grazie alla colorazione delle parole chiave e degli operatori;
- facilità d'uso;
- possibilità di inserire query e comandi SQL;
- possibilità di gestire campi variabili per l'immissione o la ricerca di dati;
- gestione di più databases;
- possibilità di salvataggio delle query più ricorrenti;

La gestione dei campi variabili è senza dubbio la caratteristica pregnante di Ezquery.

Sintassi delle istruzioni interne di Ezquery

Ezquery utilizza i tags '§!', '!§', '§.', '.§' per contraddistinguere le istruzioni interne.

SINTASSI:

1. §!nomecampo§isolatore§messaggio§query!§
2. §.nomecampo§isolatore§messaggio§query.§

   - nomecampo è il nome del campo variabile;
   - isolatore solitamente è l'apostrofo e va inserito nei campi testuali o che contengono date;
   - messaggio è il testo che si vuole mandare in output all'utente in modo che risponda interattivamente;
   - query è una query sql che produrrà in output una lista di records, dei quali, l'utente, ne dovrà scegliere uno. IMPORTANTE!!! La query deve contenere sempre una chiave primaria e questa deve essere la prima della lista dei campi.

5.2 Codice Sorgente del Software EZquery

Connessione.Open:
Sub Open()

```
  Dim f,g,h as FolderItem
  dim tos as textoutputStream
  dim tis as textInputStream
  dim rif as string
  f=GetFolderItem(":query")
  g=GetFolderItem(":settings")
  h=GetFolderItem(":settings:dbnames")
  If not(g.exists) then
    g.createAsFolder
  end if
  If not(f.exists) then
    f.createAsFolder
  end if
  If not(h.exists) then
    tos=h.CreateTextFile
    tos.close
  else
    tis=h.openAsTextFile
    do until tis.eOF
      rif = tis.ReadLine
      contextualmenu1.addRow rif
      bevelButton1.addRow rif
    loop
    tis.close
  end if
exception err as nilObjectException
  msgbox "errore durante la creazione della cartella"
End Sub

Connessione.ConnettiAlDatabase.Action:
Sub Action()
  dim test as boolean
  Dim f,g as FolderItem
  dim tos as textOutputStream
  test=connettiDB(server.text,datab.text,utente.text,password.text)
  if test then
    'aggiungo il nomeDATABASE alla lista

    'fine
    nomeDATABASE = datab.text
    f=GetFolderItem(":query:" + nomeDATABASE)
    If not(f.exists) then
      f.createAsFolder
```

```
    end if
    g=GetFolderItem(":settings:" + nomeDATABASE)
    If not(g.exists) then
      tos=g.createTextFile
      tos.close
    end if
    queryLibera.show
    connessione.close
  end if
exception err as nilObjectException
  msgbox "errore durante la creazione della cartella"
End Sub

Connessione.ContextualMenu1.Action:
Sub Action(item As String)
  datab.text = item
End Sub

Connessione.BevelButton1.Action:
Sub Action()
  datab.text =Me.list(Me.MenuValue)
End Sub

QueryLibera.Close:
Sub Close()
  connessione.show
End Sub

QueryLibera.Open:
Sub Open()
  aggiornalistaQuery
  if enable.value then
    testoquery.readOnly = false
  else
    testoquery.readOnly = true
  end if
End Sub

QueryLibera.testoquery.KeyDown:
Function KeyDown(Key As String) As Boolean
  'dim i as integer
  'if key = " " or key = chr(13) or key = chr(10) then
  'i=testoquery.selStart
```

```
  'sintassi(testoquery)
  'testoquery.selStart = i
  'end if
End Function
```

QueryLibera.OK.Action:

```
Sub Action()
  dim d as databaseCursor
  dim query, rif, errore as string
  dim bool, booltrans as boolean
  query= replaceAll(testoquery.text,"'","''")
  'controlla se è comando o query
  if instr(uppercase(query), "CREATE ") = 0 and instr(uppercase(query),
"INSERT ") = 0 and instr(uppercase(query), "DELETE ") = 0  and
instr(uppercase(query), "UPDATE ") = 0 and instr(uppercase(query),
"ALTER ") = 0 and instr(uppercase(query), "DROP ") = 0 then
    rif=controllaVariabili(query)
    d = db.SQLSelect(rif)
    if db.error then
      msgbox db.errorMessage
    else
      listaRisultati d
    end if
    d=nil
  else
    rif=controllaVariabili(query)
    db.SQLExecute rif
    errore = ""
    while not(errore = db.errormessage)
      if db.errorCode <> 1005 then
        errore = db.errormessage
        bool = confermaSi_NO(str(db.errorCode) + "  -  " +
db.errorMessage, "OK","",2)
      else
        errore = db.errormessage
      end if
    wend
    beep
  end if
end if
End Sub
```

QueryLibera.sceltaquery.Change:

```
Sub Change()
  Dim folder, file As FolderItem
  Dim path As String
  Dim fileReadFrom As TextInputStream
  dim rif as integer
  rif = sceltaquery.listIndex
  if  rif > 0 then
    sceltacomandi.listIndex = 0
    file = getfolderItem(":query:" + nomeDATABASE + ":" +
sceltaquery.Text)
    If file <> Nil then
      file.OpenStyledEditField testoquery
      sceltaquery.ListIndex = rif
    End if
  else
    testoquery.text=""
  end if
End Sub

QueryLibera.Salva.Action:
Sub Action()
  salvaconNome.showModal
End Sub

QueryLibera.B.Action:
Sub Action()
  'testoquery.ToggleSelectionBold
  sintassi(testoquery)
End Sub

QueryLibera.SceltaComandi.Change:
Sub Change()
  Dim folder, file As FolderItem
  Dim path As String
  Dim fileReadFrom As TextInputStream
  dim rif as integer
  rif = sceltaComandi.listIndex
  if  rif > 0 then
    sceltaquery.listIndex = 0
    file = getfolderItem(":query:" + nomeDATABASE + ":!" +
sceltaComandi.Text)
    If file <> Nil then
      file.OpenStyledEditField testoquery
```

```
    sceltaComandi.ListIndex = rif
   End if
  else
   testoquery.text=""
  end if
End Sub

QueryLibera.EliminaComando.Action:
Sub Action()
  dim file As FolderItem
  dim bool as boolean
  if sceltaComandi.listindex > 0 then
   bool = confermaSi_NO("VUOI DAVVERO CANCELLARE IL FILE " +
sceltaComandi.text + " ?","Cancella", "Annulla", 2)
   file = getfolderItem(":query:" + nomeDATABASE + ":!" +
sceltaComandi.Text)
   if bool then
    file.delete
   end if
  end if
  aggiornaListaQuery
End Sub

QueryLibera.EliminaQuery.Action:
Sub Action()
  dim file As FolderItem
  dim bool as boolean
  if sceltaquery.listindex > 0 then
   bool = confermaSi_NO("VUOI DAVVERO CANCELLARE IL FILE " +
sceltaquery.text + " ?","Cancella", "Annulla", 2)
   file = getfolderItem(":query:" + nomeDATABASE + ":" +
sceltaquery.Text)
   if bool then
    file.delete
   end if
  end if
  aggiornaListaQuery
End Sub

QueryLibera.AumentaTesto.Action:
Sub Action()
  testoquery.textSize = testoquery.textSize + 1
End Sub
```

```
QueryLibera.DiminuisciTesto.Action:
Sub Action()
  testoquery.textSize = testoquery.textSize - 1
End Sub

QueryLibera.Enable.Action:
Sub Action()
  if enable.value then
    testoquery.readOnly = false
  else
    testoquery.readOnly = true
  end if
End Sub

Modulo.POSTGRESQL:
Function POSTGRESQL(azione As string) As string
  Dim s As Shell
  s=New Shell
  If TargetCarbon then
    s.execute "/usr/local/bin/pg_ctl -D /usr/local/pgsql/data -l logfile
"  + azione
    If s.errorCode=0 then
      return s.result
    else
      return "Error Code: "+Str(s.errorCode)
    end if
  end if
End Function

Modulo.ConnettiDB:
Function ConnettiDB(server as string, DataBase as string, Utente as
string, Password as string) As boolean

db=OpenPostgreSQLDatabase(server,5432,database,utente,password)
  If db = Nil then
    Beep
    msgbox "Non riesco ad aprire il database. Prova ad avviare
PostgreSQL."
    return false
  else
    'msgbox "Connesso al Database"
    return true
```

```
  end if
End Function

Modulo.ListaRisultati:
Sub ListaRisultati(d as databasecursor)
  dim i, contatore as Integer
  dim v, rif as String
  result.show
  result.visible = false
  contatore = 0
  if d.eOF then
    msgBox "RecordSet vuoto"
  else
    result.ListBox1.deleteAllRows
    result.ListBox1.columncount = d.fieldcount
    for i = 0 to d.fieldcount -1
      rif = rif + "150,"
      result.ListBox1.Heading(i) = d.IdxField(i+1).Name
    next
    result.listBox1.columnWidths = mid(rif,1,len(rif)-1)
    While not d.eof
      contatore = contatore + 1
      v = d.IdxField(1).StringValue
      result.ListBox1.addrow v
      For i = 2 to d.fieldcount
        result.ListBox1.cell(result.ListBox1.lastIndex,i-
1)=d.IdxField(i).StringValue
      next
      d.MoveNext
    wend
    msgBox "Trovate " + str(contatore) + " corrispondenze!"
    result.visible = true
  end if
exception er
  result.visible = true
End Sub

Modulo.AggiornaListaQuery:
Sub AggiornaListaQuery()
  Dim i,n  as Integer
  Dim f as FolderItem
  f=getFolderItem(":query:" + nomeDATABASE)
  if f = nil then
```

```
    msgbox "Non trovo la cartella delle query per questo database."
  else
    n=f.count
    If n>0 then
      queryLibera.sceltaquery.deleteAllRows
      queryLibera.sceltaComandi.deleteAllRows
      queryLibera.sceltaquery.addRow "Scegli una query predefinita..."
      queryLibera.sceltaComandi.addRow "Scegli un comando
predefinito..."
      queryLibera.sceltaquery.listIndex = 0
      queryLibera.sceltaComandi.listIndex = 0
      For i=1 to n
        if not(mid(f.item(i).name,1,1) = ".") then 'se il file non è invisibile
          if (mid(f.item(i).name,1,1) = "!") then 'se è un comando
            queryLibera.sceltaComandi.addrow mid(f.item(i).name,2,
len(f.item(i).name)-1)
          else 'se è una query
            querylibera.sceltaquery.addRow f.item(i).name
          end if
        end if
      Next
    End if
  end if
End Sub

Modulo.ControllaVariabili:
Function ControllaVariabili(query as string) As String
  dim i, j, inizio, fine, inizio1, fine1 as integer
  dim rif, querycorretta, querycopia, nomecampo, commento,
separatore, selezione as string
  querycopia = query
  i=0
  j=0
  while instr(i,query,"§.")>0
    i = instr(i,query,"§.") + 1
    inizio = i + 1
    fine = instr(i-1,query,".§")
    rif = mid(query,inizio,fine-inizio)
    nomecampo = nthField(rif,"§",1)
    separatore = nthField(rif,"§",2)
    commento = nthField(rif,"§",3)
    selezione = nthField(rif,"§",4)
    Conferma(nomecampo, commento,"OK",1,true, selezione)
```

```
    querycorretta = nomecampo + " " + glb_operatore + " " +
separatore + glb_variabile + separatore
    j=instr(j,querycopia,"§.") + 1
    inizio1 = j + 1
    fine1 = instr(j-1,querycopia,".§")
    querycopia = mid(querycopia,1,inizio1 -3) + querycorretta +
mid(querycopia,fine1 + 2,len(querycopia)-fine1+2)
    j=0
  wend
  i=0
  j=0
  while instr(i,query,"§!")>0
    i = instr(i,query,"§!") + 1
    inizio = i + 1
    fine = instr(i-1,query,"!§")
    rif = mid(query,inizio,fine-inizio)
    nomecampo = nthField(rif,"§",1)
    separatore = nthField(rif,"§",2)
    commento = nthField(rif,"§",3)
    selezione = nthField(rif,"§",4)
    Conferma(nomecampo, commento,"OK",1, false, selezione)
    if glb_variabile <> "null" then
      querycorretta = separatore + glb_variabile + separatore
    else
      querycorretta = glb_variabile
    end if
    j=instr(j,querycopia,"§!") + 1
    inizio1 = j + 1
    fine1 = instr(j-1,querycopia,"!§")
    querycopia = mid(querycopia,1,inizio1 -3) + querycorretta +
mid(querycopia,fine1 + 2,len(querycopia)-fine1+2)
    j=0
  wend
  return querycopia
exception err
  msgbox "Errore di sintassi nella definizione di variabile"
End Function
```

Modulo.Conferma:

```
Sub Conferma(nomecampo as string, msg as string, ok as string, icon
as integer, operator as boolean, selezione as string)
  dim w as confermaFunzione
  dim d as databaseCursor
```

```
  dim db2 as database
  dim i as integer
  dim rif as string
  w = new confermaFunzione
  db2=db
  if operator then
    w.operatore.visible = true
  else
    w.operatore.visible = false
  end if
  w.msg.text = msg
  w.nomecampus.caption = nomecampo
  w.confirmWindIconNumber = icon

  if trim(selezione) = "" then
    w.scegliDa.visible = false
  else
    w.variabiles.enabled=false
    d=db2.sQLSelect(selezione)
    while not d.eOF
      rif =""
      for i = 1 to d.FieldCount
        rif = rif + d.idxField(i).StringValue + ","
      next
      rif = left(rif,len(rif)-1)
      w.scegliDa.AddRow rif
      d.MoveNext
    wend
  end if
  if ok <> "" then
    w.ok.caption = ok
  end if

  beep
  w.showmodal
End Sub

Modulo.Sintassi:
Sub Sintassi(Testo as editfield)
  dim KeyWords,rif as string
  dim KeyGreens as string
  dim i, inizio, fine as integer
```

```
  keywords = "INSERT ,UPDATE ,SET ,CREATE ,DELETE ,BETWEEN,
TABLE, FUNCTION ,FOREIGN ,CHECK , KEY ,NO ACTION, REFERENCES
,SELECT ,FROM ,PRIMARY KEY,WHERE,GROUP,BY ,HAVING,ORDER, AND
, OR , IS , NOT ,*,%, NULL, AS , DESC, ASC, DISTINCT , ON , UPDATE ,
DELETE , CASCADE, NO ACTION, NOT ,DROP , INTO , VALUES"
  keygreens = "SEQUENCE, START , VARCHAR, DEFAULT, INTEGER ,
DATE ,ON UPDATE,ON DELETE, TIMESTAMP , COUNT, NEXTVAL,
LENGTH, SUM, VARIANCE, STDDEV, MIN, MAX, AVG,DATE_PART"
  'KEYWORDS
  testo.textColor = rgb(0,0,0)
  for i = 1 to countfields(keyWords,",")
    rif=nthField(keyWords,",",i)
    inizio=0
    fine=0
    while instr(inizio, testo.text,rif) > 0
      inizio = instr(inizio, testo.text,rif) -1
      fine = inizio + len(rif)
      testo.selStart = inizio
      testo.selLength = len(rif)
      testo.selTextColor =rgb(0,0,255)
      inizio = fine + 1
    wend
  next
  'PAROLE VERDI
  for i = 1 to countfields(KeyGreens,",")
    rif=nthField(KeyGreens,",",i)
    inizio=0
    fine=0
    while instr(inizio, testo.text,rif) > 0
      inizio = instr(inizio, testo.text,rif) -1
      fine = inizio + len(rif)
      testo.selStart = inizio
      testo.selLength = len(rif)
      testo.selTextColor =rgb(0,122,0)
      inizio = fine + 1
    wend
  next
  'PAROLE ROSSE
  inizio = 0
  fine = 0
  while instr(inizio, testo.text,"§.") > 0
    inizio = instr(inizio, testo.text,"§.")
    while instr(inizio, testo.text,".§") > 0
```

```
    fine=instr(inizio, testo.text,".§")
    testo.selStart = inizio - 2
    testo.selLength = (fine - inizio) +3
    testo.selTextColor =rgb(200,0,0)
    inizio = fine + 1
   wend
  wend
  inizio = 0
  fine = 0
  while instr(inizio, testo.text,"§!") > 0
   inizio = instr(inizio, testo.text,"§!")
   while instr(inizio, testo.text,"!§") > 0
    fine=instr(inizio, testo.text,"!§")
    testo.selStart = inizio - 2
    testo.selLength = (fine - inizio) +3
    testo.selTextColor =rgb(200,10,0)
    inizio = fine + 1
   wend
  wend
End Sub

Modulo.ConfermaSi_NO:
Function ConfermaSi_NO(msg as string, ok as string, canceltxt as
string, icon as integer) As boolean
  dim w as ConfirmFuncWind
  w = new ConfirmFuncWind

  w.msg.text = msg
  w.confirmWindIconNumber = icon

  if canceltxt <> "" then
   w.cancelbtn.caption = canceltxt
  end if
  if ok <> "" then
   w.ok.caption = ok
  end if

  beep
  w.showmodal

  return confirmReturnBoolean
End Function
```

```
App.EnableMenuItems:
Sub EnableMenuItems()
  //**********
  postgreSQLAvvia.enable
  postgreSQLriavvia.enable
  postgreSQLchiudi.enable
  //**********
  fileQuit.enable
  fileSalva.enable
  //**********
  editClear.enable
  editCopy.enable
  editCut.enable
  editPaste.enable
  editUndo.enable
  editSelectAll.enable
  //**********
End Sub

Result.PushButton1.Action:
Sub Action()
  result.close
End Sub

Result.ScrollBar1.ValueChanged:
Sub ValueChanged()
  dim i as integer
  dim rif as string
  for i = 0 to listBox1.columnCount
    rif = rif + str(scrollBar1.value) + ","
  next
  listBox1.columnWidths = left(rif,len(rif)-1)
  'str(scrollBar1.value)
End Sub

Result.EsportaDati.Action:
Sub Action()
  Dim file As FolderItem
  Dim fileStream As TextOutputStream
  dim i,j as integer
  dim rif as string
  file=GetSaveFolderItem("application/text","esportazione dati")
  fileStream=file.CreateTextFile
```

```
  for i= 0 to listBox1.listCount -1
    rif = ""
    for j = 0 to listBox1.columnCount -1
      rif = rif + listBox1.cell(i,j) + ";"
    next
    fileStream.WriteLine mid(rif,1, len(rif)-1)
  next
  fileStream.Close
exception err
  msgbox "si è verificato un errore sconosciuto"
End Sub

ConfermaFunzione.iconCanvas.Paint:
Sub Paint(g As Graphics)
  select case confirmWindIconNumber
  case 1
    Me.Graphics.DrawNoteIcon 0,0
  case 2
    Me.Graphics.DrawCautionIcon 0,0
  else
    Me.Graphics.DrawStopIcon 0,0
  end
End Sub

ConfermaFunzione.ok.Action:
Sub Action()
  if variabiles.enabled = false then
    if nthField(scegliDa.text,",",1) = "" then
      glb_variabile = "null"
    else
      glb_variabile = nthField(scegliDa.text,",",1)
    end if
  else
    if trim(variabiles.text) = "" then
      glb_variabile = "null"
    else
      glb_variabile = uppercase(variabiles.text)
    end if
  end if
  glb_operatore = operatore.text
  self.close
End Sub
```

SalvaConNome.Open:

```
Sub Open()
  dim rif as string
  rif = queryLibera.testoquery.text
  if queryLibera.sceltaquery.listIndex > 0 then
    nomedelfile.text = queryLibera.sceltaquery.text
  end if
  if queryLibera.sceltacomandi.listIndex > 0 then
    nomedelfile.text = queryLibera.sceltacomandi.text
  end if
End Sub
```

SalvaConNome.Salva.Action:

```
Sub Action()
  Dim file as folderItem
  Dim fileStream As TextOutputStream
  dim rif, fileorigine as string
  dim scelta1, scelta2 as integer
  rif = queryLibera.testoquery.text
  if queryLibera.sceltaquery.listIndex > 0 then
    FileOrigine = queryLibera.sceltaquery.text
    scelta1=queryLibera.sceltaquery.listIndex
    scelta2 = -1
  end if
  if queryLibera.sceltacomandi.listIndex > 0 then
    FileOrigine = "!" + queryLibera.sceltacomandi.text
    scelta1= -1
    scelta2 = queryLibera.sceltacomandi.listIndex
  end if
  if rinomina.value then
    file=getFolderItem(":query:" + nomeDATABASE + ":" + fileorigine)
    if file <> nil then
      if instr(uppercase(rif), "CREATE ") = 0 and instr(uppercase(rif),
"INSERT ") = 0 and instr(uppercase(rif), "DELETE ") = 0  and
instr(uppercase(rif), "UPDATE ") = 0 and instr(uppercase(rif), "ALTER ")
= 0 then
        file.name = nomedelfile.text
      else
        file.name = "!" + nomedelfile.text
      end if
      sintassi querylibera.testoquery
      file.saveStyledEditField querylibera.testoquery
    end if
```

```
  else
    if instr(uppercase(rif), "CREATE ") = 0 and instr(uppercase(rif),
"INSERT ") = 0 and instr(uppercase(rif), "DELETE ") = 0  and
instr(uppercase(rif), "UPDATE ") = 0 and instr(uppercase(rif), "ALTER ")
= 0 then
      file=getFolderItem(":query:" + nomeDATABASE + ":" +
nomedelfile.text)
    else
      file=getFolderItem(":query:" + nomeDATABASE + ":!" +
nomedelfile.text)
    end if
    sintassi querylibera.testoquery
    file.saveStyledEditField querylibera.testoquery
  end if
  aggiornaListaQuery
  if scelta1 = -1 then
    queryLibera.sceltacomandi.listIndex = scelta2
  else
    queryLibera.sceltaquery.listIndex = scelta1
  end if
  salvaconNome.close
End Sub

SalvaConNome.Annulla.Action:
Sub Action()
  salvaConNome.close
End Sub

ConfirmFuncWind.iconCanvas.Paint:
Sub Paint(g As Graphics)
  select case confirmWindIconNumber
  case 1
    Me.Graphics.DrawNoteIcon 0,0
  case 2
    Me.Graphics.DrawCautionIcon 0,0
  else
    Me.Graphics.DrawStopIcon 0,0
  end
End Sub

ConfirmFuncWind.ok.Action:
Sub Action()
  confirmReturnBoolean = true
```

```
  self.close
End Sub

ConfirmFuncWind.cancelbtn.Action:
Sub Action()
  confirmReturnBoolean = false
  self.close
End Sub
```

VI – Dati di Esempio

```
Welcome to Darwin!
[localhost:~] giovanni% psql softwarehouse
Welcome to psql, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit

softwarehouse=# select * from progetto;
 id |   nome   | descrizione | data_inizio | data_debugging | data_rilascio |
data_prevista
----+----------+-------------+-------------+----------------+---------------+---
------------
  1 | MAC OS X | IL NUOVO OS | 2002-03-11  |                |               |
2002-04-11
(1 row)

softwarehouse=# select * from applicazione;
 id |  nome  |            descrizione             | data_rilascio | linguaggio |
data_inizio | data_debugging | data_prevista |    ruolo     | id_progetto
----+--------+------------------------------------+---------------+------------
+-------------+----------------+---------------+--------------+-------------
  1 | DOCK   | BARRA PRESENTE IN FONDO AL DESKTOP |               | C++        |
2002-03-11  |                |               | FILE MANAGER |           1
  2 | AQUA   | INTERFACCIA GRAFICA                |               | PDF        |
2002-03-11  |                |               | GUI          |           1
  3 | DARWIN | TERMINALE UNIX                     | 2002-03-12    | C++        |
2002-03-11  |                |               | TERMINALE    |           1
(3 rows)

softwarehouse=# select * from intervento;
 id | id_applicazione |      descrizione       | tipo |    data
----+-----------------+------------------------+------+------------
  1 |               3 | FISSATI BUGS DI STAMPA |    1 | 2002-03-11
  2 |               3 | RELEASE 0.1            |    2 | 2002-03-11
(2 rows)

softwarehouse=# select * from qualifica_necessaria;
 id | quantita | id_applicazione |       id_qualifica
----+----------+-----------------+-------------------------
  6 |        1 |               1 | PROGRAMMATORE JAVASCRIPT
  7 |        1 |               2 | PROGRAMMATORE JAVASCRIPT
  8 |        2 |               2 | WEB MASTER
  9 |        1 |               3 | PROGRAMMATORE JAVASCRIPT
 10 |        1 |               3 | PROGRAMMATORE ASP
(5 rows)

softwarehouse=# select * from impegno;
 id_qualifica_necessaria |  id_lavoratore   | data_inizio | data_fine  |
compenso | progetto
-------------------------+------------------+-------------+------------+--------
--+----------
                       6 | LMBDNL81D30I838Z | 2002-03-11  |            |
|        1
```

```
                        8 | DNTCRM81N09I838Z | 2002-03-08  | 2002-03-10 |
|        1
                        9 | DNTCRM81N09I838Z | 2002-01-01  | 2002-01-05 |
|        1
(3 rows)

softwarehouse=# select * from lavoratore;
   cod_fiscale    |  cognome  |   nome
------------------+-----------+----------
 GSPGNN81D30I838Z | GASPARRI  | GIOVANNI
 DNTCRM81N09I838Z | DI NATALE | CARMINE
 PLLPLA81D30I838Z | PALLESCHI | PAOLO
 LMBDNL81D30I838Z | LOMBARDI  | DANILO
(4 rows)

softwarehouse=# select * from impiegato;
   cod_fiscale    | data_assunzione | stipendio | matricola
------------------+-----------------+-----------+-----------
 DNTCRM81N09I838Z | 2002-01-01      |   1500.38 |        10
 LMBDNL81D30I838Z | 2002-01-01      |   1500.60 |        20
 PLLPLA81D30I838Z | 2002-01-01      |   1200.32 |        30
(3 rows)

softwarehouse=# select * from esecuzione;
 id_intervento |  id_lavoratore
---------------+------------------
             1 | GSPGNN81D30I838Z
             2 | PLLPLA81D30I838Z
(2 rows)

softwarehouse=# select * from intervento;
 id | id_applicazione |      descrizione       | tipo |    data
----+-----------------+------------------------+------+------------
  1 |               3 | FISSATI BUGS DI STAMPA |    1 | 2002-03-11
  2 |               3 | RELEASE 0.1            |    2 | 2002-03-11
(2 rows)

softwarehouse=# select * from partecipazione;
 id_corso |  id_lavoratore   | docente
----------+------------------+---------
        1 | DNTCRM81N09I838Z | f
        1 | LMBDNL81D30I838Z | f
(2 rows)

softwarehouse=# select * from corso;
 id |         nome         | ore | data_inizio | data_fine
----+----------------------+-----+-------------+------------
  1 | MACROMEDIA ROAD SHOW |   5 | 2002-01-01  | 2002-01-01
(1 row)

softwarehouse=# select * from abilitazione;
 id_corso |       id_qualifica
----------+--------------------------
        1 | PROGRAMMATORE ASP
        1 | PROGRAMMATORE JAVASCRIPT
        1 | WEB MASTER
(3 rows)

softwarehouse=# select * from qualifica;
          nome
-------------------------
 PROGRAMMATORE JAVA
 PROGRAMMATORE C++
```

```
 GRAFICO
 WEB MASTER
 PROGRAMMATORE ASP
 PROGRAMMATORE PERL
 PROGRAMMATORE JAVASCRIPT
(7 rows)

softwarehouse=# select * from possesso;
  id_lavoratore   |       id_qualifica       | corso
------------------+--------------------------+-------
 DNTCRM81N09I838Z | PROGRAMMATORE ASP        |     1
 DNTCRM81N09I838Z | PROGRAMMATORE JAVASCRIPT |     1
 DNTCRM81N09I838Z | WEB MASTER               |     1
 LMBDNL81D30I838Z | PROGRAMMATORE ASP        |     1
 LMBDNL81D30I838Z | PROGRAMMATORE JAVASCRIPT |     1
 LMBDNL81D30I838Z | WEB MASTER               |     1
 GSPGNN81D30I838Z | PROGRAMMATORE ASP        |
(7 rows)

softwarehouse=
```