|  |  |
| --- | --- |
|  |  |

Basi di Dati e Conoscenza

Progetto A.A. 2019/2020

Sistema per la gestione della vendita all’ingrosso di piante

0252795

Simone Tiberi

**Indice**

[1. Descrizione del Minimondo 2](#_Toc606296459)

[2. Analisi dei Requisiti 3](#_Toc1289394997)

[3. Progettazione concettuale 9](#_Toc2081466291)

[4. Progettazione logica 17](#_Toc2147004904)

[5. Progettazione fisica 35](#_Toc518560220)

[Appendice: Implementazione 114](#_Toc403811585)

# Descrizione del Minimondo

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7  8 9 10 11 12 13 14 15 16 17  18 19 20 21 22 23  24 25 26 27 28 29  30 31 32  33 34 35 36 37 38 | L’azienda Verde S.r.l. gestisce la vendita all’ingrosso di piante da interni ed esterni. L’azienda tratta diverse specie di piante, ciascuna caratterizzata sia dal nome latino che dal nome comune, e da un codice univoco alfanumerico attraverso cui la specie viene identificata. Per ciascuna specie è inoltre noto se sia tipicamente da giardino o da appartamento e se sia una specie esotica o meno. Le piante possono essere verdi oppure fiorite. Nel caso di specie di piante fiorite, sono note tutte le colorazioni in cui una specie è disponibile.  L’azienda gestisce ordini massivi ed ha un parco clienti sia di rivendite che di privati. Per ciascun privato sono noti il codice fiscale, il nome e l’indirizzo della persona, mentre per ogni rivendita sono noti la partita iva, il nome e l’indirizzo della rivendita. In entrambi i casi, è possibile mantenere un numero arbitrario di contatti, ad esempio numeri di telefono, di cellulare, di indirizzi email. Per ciascun cliente è possibile indicare qual è il mezzo di comunicazione preferito per essere contattati. Nel caso di una rivendita, è necessario mantenere anche il nome/cognome di un referente, eventualmente associato ad altri contatti (con la possibilità, sempre, di indicarne uno preferito). Sia i clienti privati che le rivendite devono avere un indirizzo di fatturazione, che può essere differente dall’indirizzo di residenza o dall’indirizzo di spedizione.  I fornitori di Verde S.r.l. sono identificati attraverso un codice fornitore; per ciascun fornitore sono inoltre noti il nome, il codice fiscale ed un numero arbitrario di indirizzi. Il fornitore può fornire diverse specie di piante. Verde S.r.l. ha un dipartimento di gestione di magazzino che tiene traccia delle giacenze ed effettua, periodicamente, ordini ai fornitori per mantenere una giacenza di tutte le specie di piante trattate. Le specie di piante trattate sono gestite dai manager di Verde S.r.l.  Si vuole tener traccia di tutti gli acquisti eseguiti da ciascun cliente. Un acquisto, effettuato in una data specifica, è relativo a una certa quantità di piante appartenenti ad un certo numero di specie. Nell’ambito di un ordine è di interesse sapere a quale indirizzo questo deve essere inviato, e quale referente (se presente) e quale contatto fornire al corriere per mettersi in contatto con il destinatario in caso di problemi nella consegna. Non è possibile aprire un ordine se non vi è disponibilità in magazzino.  Il listino prezzi, in cui si vuole tener traccia dei prezzi assunti nel tempo da ciascuna specie di piante. Una variazione di prezzo non deve avere effetto su un ordine già aperto ma non ancora finalizzato. I prezzi sono gestiti dai manager di Verde S.r.l.  Gli ordini vengono evasi in pacchi. Un ordine è associato ad un numero arbitrario di pacchi ed è di interesse di Verde S.r.l. tenere traccia di quali piante sono contenute all’interno di un pacco. Per motivi di ottimizzazione degli spazi, un pacco può contenere un insieme differente di specie di piante. Quando si prepara un pacco, è di interesse per l’operatore sapere quali piante devono essere ancora inserite nei pacchi, al fine di evadere correttamente l’ordine. |

# Analisi dei Requisiti

## Identificazione dei termini ambigui e correzioni possibili

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Linea** | | **Termine** | | **Nuovo termine** | | **Motivo correzione** | |
| 4 | | giardino | | esterni | | Il termine *esterni* si adatta meglio al minimondo di interesse. | |
| 5 | | appartamento | | interni | | Il termine *interni* si adatta meglio al minimondo di interesse. | |
| 9 | | persona | | privato | | Il termine *privato* rappresenta meglio il concetto duale di *rivendita* nel minimondo di interesse. | |
| 9 | | indirizzo della persona | | indirizzo di residenza | | Come si capisce dalla specifica ulteriore a riga 16-17, i due indirizzi corrispondono.  Inoltre il termine *residenza* meglio rappresenta un concetto generico del cliente anziché della sua specializzazione persona. | |
| 10 | | indirizzo della rivendita | | indirizzo di residenza | | Come si capisce dalla specifica ulteriore a riga 16-17, i due indirizzi corrispondono.  Inoltre il termine *residenza* meglio rappresenta un concetto generico del cliente anziché della sua specializzazione rivendita. | |
| 12 | | mezzo di comunicazione | | contatto | | La frase “eventualmente associato ad altri contatti (con la possibilità, sempre, di indicarne uno preferito).” (righe 14-15) lascia intendere che i due termini sono effettivamente sinonimi. | |
| 21 | | ordine | | richiesta di fornitura | | Il termine *ordine* verrà utilizzato nella specifica disambiguata per ulteriori scopi. | |
| 24 | | acquisti | | ordine | | Il termine *ordine* è stato scelto per rappresentare l’acquisto poiché più semanticamente corretto nelle diverse frasi in cui compare. | |
| 28 | | destinatario | | cliente | | Il termine *cliente* è già presente nella specifica.  Il termine *destinatario* non viene utilizzato altrove nella specifica ed è meno adatto a rappresentare il concetto. | |

### Specifica disambiguata

|  |
| --- |
| L’azienda Verde S.r.l. gestisce la vendita all’ingrosso di piante da interni ed esterni. L’azienda tratta diverse specie di piante, ciascuna caratterizzata sia dal nome latino che dal nome comune, e da un codice univoco alfanumerico attraverso cui la specie viene identificata. Per ciascuna specie è inoltre noto se sia tipicamente da esterni o da interni e se sia una specie esotica o meno. Le piante possono essere verdi oppure fiorite. Nel caso di specie di piante fiorite, sono note tutte le colorazioni in cui una specie è disponibile.  L’azienda gestisce ordini massivi ed ha un parco clienti sia di rivendite che di privati. Per ciascun privato sono noti il codice fiscale, il nome e l’indirizzo di residenza, mentre per ogni rivendita sono noti la partita iva, il nome e l’indirizzo di residenza. In entrambi i casi, è possibile mantenere un numero arbitrario di contatti, ad esempio numeri di telefono, di cellulare, di indirizzi email. Per ciascun cliente è possibile indicare qual è il contatto preferito per essere contattati. Nel caso di una rivendita, è necessario mantenere anche il nome/cognome di un referente, eventualmente associato ad altri contatti (con la possibilità, sempre, di indicarne uno preferito). Sia i clienti privati che le rivendite devono avere un indirizzo di fatturazione, che può essere differente dall’indirizzo di residenza o dall’indirizzo di spedizione.  I fornitori di Verde S.r.l. sono identificati attraverso un codice fornitore; per ciascun fornitore sono inoltre noti il nome, il codice fiscale ed un numero arbitrario di indirizzi. Il fornitore può fornire diverse specie di piante. Verde S.r.l. ha un dipartimento di gestione di magazzino che tiene traccia delle giacenze ed effettua, periodicamente, richieste di forniture ai fornitori per mantenere una giacenza di tutte le specie di piante trattate. Le specie di piante trattate sono gestite dai manager di Verde S.r.l.  Si vuole tener traccia di tutti gli ordini effettuati da ciascun cliente. Un ordine, effettuato in una data specifica, è relativo a una certa quantità di piante appartenenti ad un certo numero di specie. Nell’ambito di un ordine è di interesse sapere a quale indirizzo questo deve essere inviato, e quale referente (se presente) e quale contatto fornire al corriere per mettersi in contatto con il cliente in caso di problemi nella consegna. Non è possibile aprire un ordine se non vi è disponibilità in magazzino.  Nel listino prezzi si vuole tener traccia dei prezzi assunti nel tempo da ciascuna specie di piante. Una variazione di prezzo non deve avere effetto su un ordine già aperto ma non ancora finalizzato. I prezzi sono gestiti dai manager di Verde S.r.l.  Gli ordini vengono evasi in pacchi. Un ordine è associato ad un numero arbitrario di pacchi ed è di interesse di Verde S.r.l. tenere traccia di quali piante sono contenute all’interno di un pacco. Per motivi di ottimizzazione degli spazi, un pacco può contenere un insieme differente di specie di piante. Quando si prepara un pacco, è di interesse per l’operatore sapere quali piante devono essere ancora inserite nei pacchi, al fine di evadere correttamente l’ordine. |

## Glossario dei Termini

|  |  |  |  |
| --- | --- | --- | --- |
| **Termine** | **Descrizione** | **Sinonimi** | **Collegamenti** |
| Specie di pianta | Bene venduto dalla Verde S.r.l.  Classificazione:   * Da interni * Da esterni * Verde * Fiorita   Può infine essere esotica. |  | Pacco, Fornitore ed Ordine |
| Cliente | Acquirente dei beni venduti dalla Verde S.r.l. |  | Ordine e Contatto |
| Privato | Possibile tipologia di cliente della Verde S.r.l |  |  |
| Rivendita | Possibile tipologia di cliente della Verde S.r.l |  | Referente |
| Referente | Persona di riferimento per una rivendita nei confronti di Verde S.r.l |  | Contatto, Rivendita, Ordine |
| Fornitore | Ente che fornisce piante alla Verde S.r.l. |  | Specie di pianta |
| Ordine | Richiesta di acquisto piante da parte dei clienti di Verde S.r.l. |  | Specie di pianta, Contatto, Pacco e Cliente, Referente |
| Pacco | Elemento che permette la suddivisione delle piante da inoltrare nell’ordine da evadere |  | Ordine e Specie di pianta |
| Manager | Dipendente della Verde S.r.l incaricato della gestione delle specie vendute e i loro prezzi. |  | Specie di pianta |
| Addetto dipartimento magazzino | Dipendente della Verde S.r.l. incaricato di richiedere la fornitura di piante per mantenerne la giacenza |  | Specie di pianta e Fornitore |
| Operatore pacchi | Dipendente della Verde S.r.l. incaricato della lavorazione dell’ordine |  | Specie di pianta, Pacco, Ordine |

## Raggruppamento dei requisiti in insiemi omogenei

|  |
| --- |
| **Frasi relative a *specie di pianta*** |
| L’azienda Verde S.r.l. gestisce la vendita all’ingrosso di piante da interni ed esterni. L’azienda tratta diverse piante, ciascuna caratterizzata sia dal nome latino che dal nome comune, e da un codice univoco alfanumerico attraverso cui viene identificata. Per ciascuna pianta è inoltre noto se sia tipicamente da interni o da esterni e se sia una pianta esotica o meno. Le piante possono essere verdi oppure fiorite. Nel caso di piante fiorite, sono note tutte le colorazioni in cui una pianta è disponibile. |
| Il fornitore può fornire diverse piante. Verde S.r.l. ha un dipartimento di gestione di magazzino che tiene traccia delle giacenze ed effettua, periodicamente, richieste di fornitura ai fornitori per mantenere una giacenza di tutte le piante trattate. Le piante trattate sono gestite dai manager di Verde S.r.l. |
| Un ordine, effettuato in una data specifica, è relativo a una certa quantità di piante. |
| Nel listino prezzi si vuole tener traccia dei prezzi assunti nel tempo da ciascuna pianta. Una variazione di prezzo non deve avere effetto su un ordine già aperto ma non ancora finalizzato. I prezzi sono gestiti dai manager di Verde S.r.l. |
| è di interesse di Verde S.r.l. tenere traccia di quali piante sono contenute all’interno di un pacco. Per motivi di ottimizzazione degli spazi, un pacco può contenere un insieme di piante differenti. Quando si prepara un pacco, è di interesse per l’operatore sapere quali piante devono essere ancora inserite nei pacchi, al fine di evadere correttamente l’ordine. |
| Nel listino prezzi si vuole tener traccia dei prezzi assunti nel tempo da ciascuna specie di piante. |

|  |
| --- |
| **Frasi relative a *cliente*** |
| L’azienda gestisce ordini massivi ed ha un parco clienti sia di rivendite che di privati. |
| In entrambi i casi, è possibile mantenere un numero arbitrario di contatti, ad esempio numeri di telefono, di cellulare, di indirizzi email. Per ciascun cliente è possibile mantenere un numero arbitrario di contatti, ad esempio numeri di telefono, di cellulare, di indirizzi email ed è possibile indicare qual è il contatto preferito per essere contattati. |
| Sia i clienti privati che le rivendite devono avere un indirizzo di fatturazione, che può essere differente dall’indirizzo di residenza o dall’indirizzo di spedizione. |
| Si vuole tener traccia di tutti gli ordini eseguiti da ciascun cliente. |
| Nell’ambito di un ordine è di interesse sapere […] quale contatto fornire al corriere per mettersi in contatto con il cliente in caso di problemi nella consegna. |

|  |
| --- |
| **Frasi relative a *privato*** |
| Per ciascun privato sono noti il codice fiscale, il nome e l’indirizzo di residenza. |

|  |
| --- |
| **Frasi relative a *rivendita*** |
| Per ogni rivendita sono noti la partita iva, il nome e l’indirizzo di residenza. |
| Nel caso di una rivendita, è necessario mantenere anche il nome/cognome di un referente, eventualmente associato ad altri contatti (con la possibilità, sempre, di indicarne uno preferito) |

|  |
| --- |
| **Frasi relative a *referente*** |
| Nel caso di una rivendita, è necessario mantenere anche il nome/cognome di un referente, eventualmente associato ad altri contatti (con la possibilità, sempre, di indicarne uno preferito). |
| Nell’ambito di un ordine è di interesse sapere a quale indirizzo questo deve essere inviato, e quale referente (se presente) e quale contatto fornire al corriere per mettersi in contatto con il cliente in caso di problemi nella consegna. |

|  |
| --- |
| **Frasi relative a *fornitore*** |
| I fornitori di Verde S.r.l. sono identificati attraverso un codice fornitore; per ciascun fornitore sono inoltre noti il nome, il codice fiscale ed un numero arbitrario di indirizzi. Il fornitore può fornire diverse specie di piante. Verde S.r.l. ha un dipartimento di gestione di magazzino che tiene traccia delle giacenze ed effettua, periodicamente, richiesta di fornitura ai fornitori per mantenere una giacenza di tutte le specie di piante trattate. |

|  |
| --- |
| **Frasi relative a *ordine*** |
| Si vuole tener traccia di tutti gli ordini effettuati da ciascun cliente. Un ordine, effettuato in una data specifica, è relativo a una certa quantità di piante appartenenti ad un certo numero di specie. Nell’ambito di un ordine è di interesse sapere a quale indirizzo questo deve essere inviato, e quale referente (se presente) e quale contatto fornire al corriere per mettersi in contatto con il cliente in caso di problemi nella consegna. Non è possibile aprire un ordine se non vi è disponibilità in magazzino. |
| Una variazione di prezzo non deve avere effetto su un ordine già aperto ma non ancora finalizzato. |
| Gli ordini vengono evasi in pacchi. Un ordine è associato ad un numero arbitrario di pacchi. |

|  |
| --- |
| **Frasi relative a *pacco*** |
| Gli ordini vengono evasi in pacchi. Un ordine è associato ad un numero arbitrario di pacchi ed è di interesse di Verde S.r.l. tenere traccia di quali piante sono contenute all’interno di un pacco. Per motivi di ottimizzazione degli spazi, un pacco può contenere un insieme differente di specie di piante. Quando si prepara un pacco, è di interesse per l’operatore sapere quali piante devono essere ancora inserite nei pacchi, al fine di evadere correttamente l’ordine. |

|  |
| --- |
| **Frasi relative a *manager*** |
| Le specie di piante trattate sono gestite dai manager di Verde S.r.l. |
| I prezzi sono gestiti dai manager di Verde S.r.l. |

|  |
| --- |
| **Frasi relative a *addetto* *dipartimento magazzino*** |
| Verde S.r.l. ha un dipartimento di gestione di magazzino che tiene traccia delle giacenze ed effettua, periodicamente, richiesta di fornitura ai fornitori per mantenere una giacenza di tutte le specie di piante trattate. |

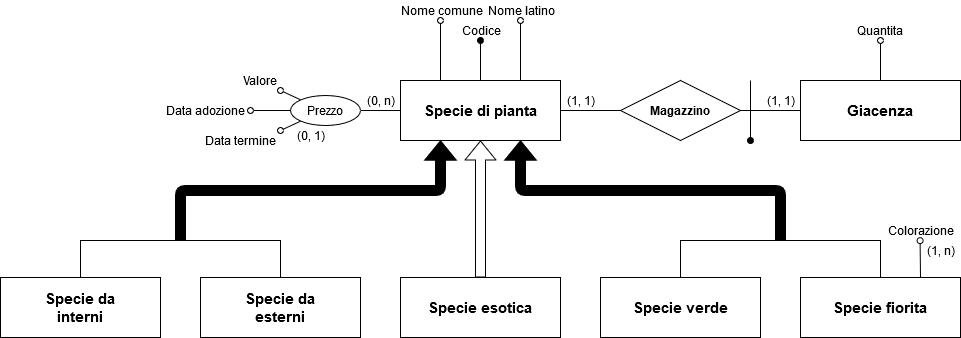
|  |
| --- |
| **Frasi relative a *operatore pacchi*** |
| Quando si prepara un pacco, è di interesse per l’operatore sapere quali piante devono essere ancora inserite nei pacchi, al fine di evadere correttamente l’ordine. |

# Progettazione concettuale

## Costruzione dello schema E-R

Per la realizzazione dello schema E-R definitivo è stato seguito un approccio bottom-up i cui passi sono qui sotto riportati.

La modellazione delle entità durante la prima fase di analisi ha prodotto i seguenti schemi parziali:

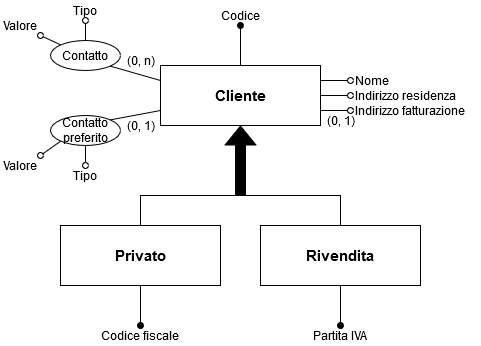


L’entità *specie di pianta* è stata sin da subito modellata mediante generalizzazioni totali e parziali per caratterizzare al meglio la classificazione evidenziata nella prima delle frasi nella sezione relativa ad essa.

È stato introdotto un attributo composto multivalore per rappresentare il prezzo, poiché come si evince nel penultimo paragrafo della specifica, tale necessità delle informazioni relative a data adozione e data termine di utilizzo.

Inoltre ai fini del progetto non è di interesse tenere traccia della singola pianta bensì soltanto della quantità per ciascuna specie motivo per cui è stata introdotta l’entità *giacenza* debole rispetto a *specie di pianta*.

In questa fase progettuale è stato scelto di scindere le entità *specie* *di pianta* e *giacenza* poiché intrinsecamente distinte (il primo infatti caratterizza un concetto astratto di classificazione il secondo la situazione attuale concreta del magazzino relativamente alla prima).

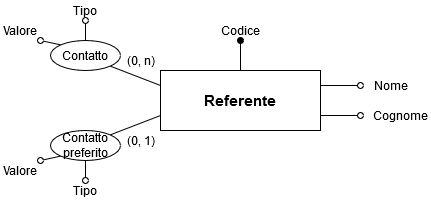


L’entità *cliente* è stata sin da subito modellata mediante generalizzazione totale per catturare al meglio la distinzione fra *privati* e *rivendite*.

Nel testo non è indicato un identificativo univoco per il *cliente* (entità padre) per questo motivo ne è stato introdotto uno ad-hoc.

Il testo suggerisce però esplicitamente degli identificativi univoci per le entità figlie (ovvero il codice fiscale e la partita iva).

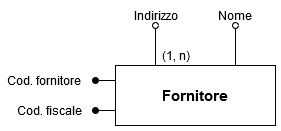
Inoltre l’indirizzo di fatturazione dato che può essere differente o meno dall’indirizzo di residenza è stato modellato come attributo opzionale.

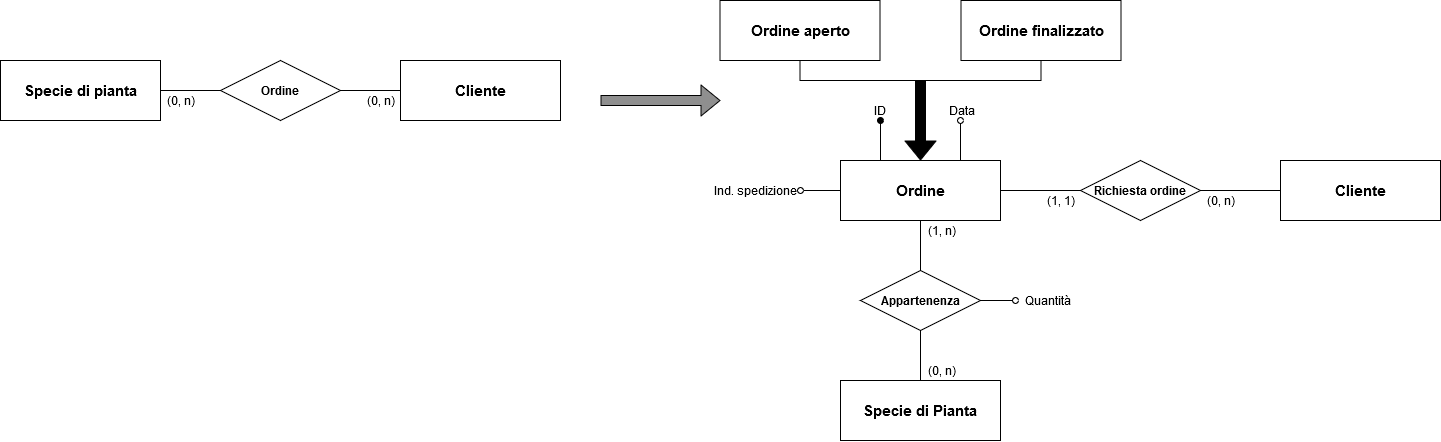


Seppure il concetto di *referente* non sia particolarmente enfatizzato all’interno del testo, sin dall’inizio dell’analisi è stato deciso di reificarlo ad entità mediante un processo di partizionamento verticale.

Tale decisione è stata presa al fine di evitare l’utilizzo di un attributo composto dell’entità *cliente* di elevata complessità per rappresentare il *referente* (di fatti avrebbe contenuto a sua volta una coppia di ulteriori attributi composti per la gestione dei contatti).

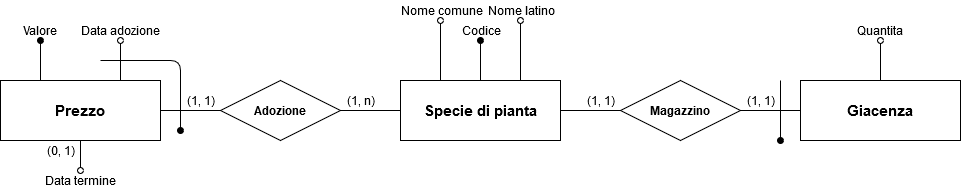
Essendo nata da un processo di partizionamento è stato scelto di identificare l’entità *referente* debolmente mediante l’identificatore esterno di cliente.



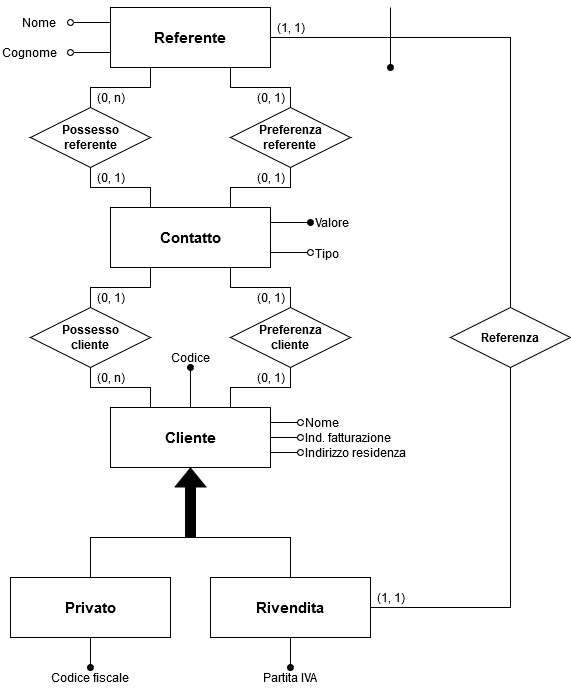
Per quanto riguarda l’entità *fornitore* la specifica fornita suggerisce due identificativi possibili entrambi riportati nello schema parziale qui riportato.

Inizialmente l’entità *ordine* è stata modellata come associazione m:n fra le entità *cliente* e *specie di pianta*. Tale è stata poi reificata ad entità per far si che un cliente potesse effettuare più ordini a cui afferisse la medesima specie di pianta. In questo scenario è stato aggiunto un attributo identificatore ad hoc per l’entità (La scelta non è ricaduta sull’identificazione mediante attributo esterno per una sorta di continuità con le altre piattaforme già esistenti in cui solitamente ciascun ordine è identificato da un codice numerico).

Per garantire infine la distinzione fra *ordine aperto* e *finalizzato* infineè stata quindi introdotta una generalizzazione totale.

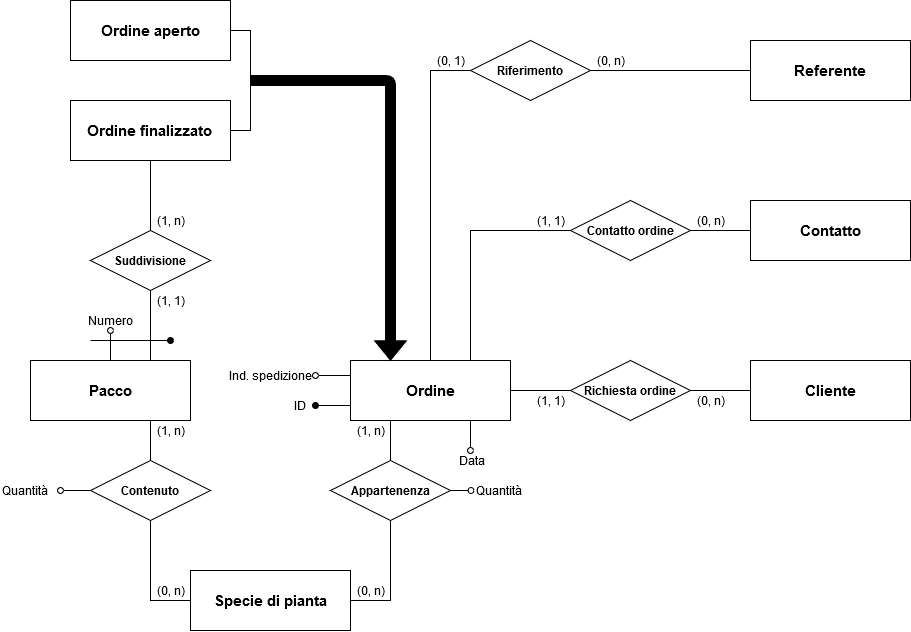
In una seconda fase del processo di analisi sono state individuate le seguenti decomposizioni a partire dagli schemi sopra presentati:

Sull’attributo composto multivalore prezzodell’entità *specie di pianta* è stato attuato un processo di reificazione di attributo di entità, come è evidenziato nello schema parziale sopra riportato.



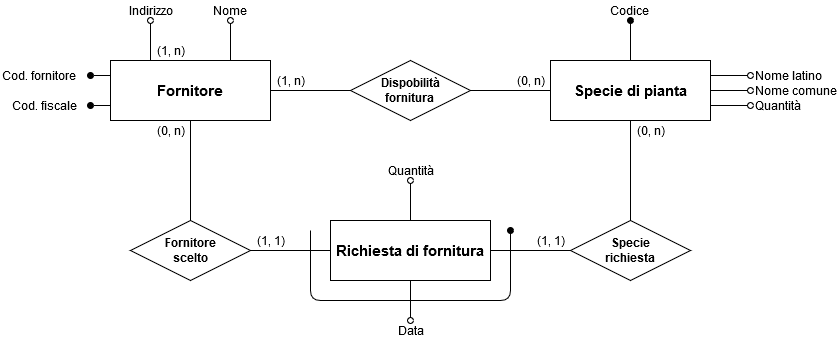
Anche l’attributo contatto delle entità *cliente* e *referente* ha subito lo stesso processo di reificazione.

In particolare la neonata entità *contatto* è stata collegata alle altre due entità mediante due coppie di associazioni distinte per mantenere la distinzione fra contatto e contatto preferito come nell’analisi precedentemente effettuata.



Lo schema parziale qui sopra riportato è relativo la struttura del core del diagramma, ovvero dell’insieme delle associazioni che legano i vari elementi che sono stati precedentemente descritti singolarmente.

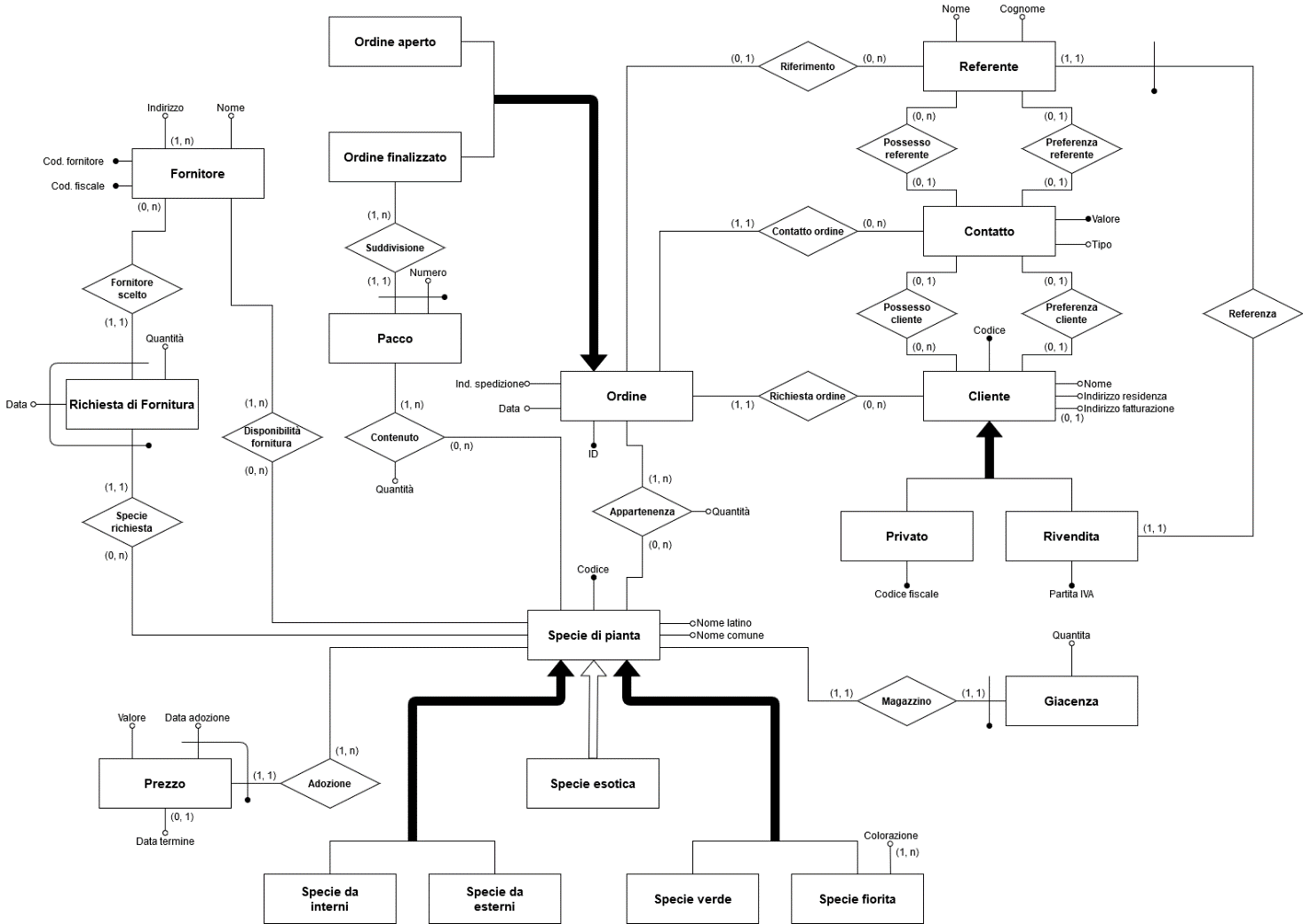
A tal proposito è stata introdotta l’entità *pacco* (debole rispetto ad *ordine*) identificata da un numero progressivo unico all’interno dell’*ordine* stesso. Tale entità è stata introdotta al fine di discriminare l’appartenenza di una specie ad un ordine e l’appartenenza di una pianta ad un pacco.



Nello schema parziale presentato qui sopra è evidenziata la scelta di collegare le entità *fornitore* e *specie di pianta* in due modi distinti:

* mediante una associazione m:n per evidenziare il fatto che un fornitorepossiede una determinata specie (ovvero può rifornire Verde S.r.l.)
* reificando una associazione m:n fra i due per far si che un fornitore possa fornire la stessa specie diverse volte.

### Integrazione finale

Non essendovi conflitti da risolvere di seguito è direttamente riportato lo schema definitivo.

## Regole aziendali

RV01) Il prezzo attualmente adottato DEVE essere l’unico avente *data termine* nulla.

RV02) La *data* di *adozione* di un prezzo DEVE essere antecedente alla *data* di *termine.*

RV03) Il contatto preferito DEVE appartenere all’insieme dei contatti posseduti.

RV04) La quantità di piante di una specie richiesta in un ordine DEVE essere minore o uguale della giacenza della stessa.

RV05) La quantità di piante di una specie contenuta in un pacco DEVE esser minore o uguale del numero di piante della specie selezionate per l’ordine a cui esso afferisce.

RV06) Il referente associato all’ordine, se esiste, DEVE corrispondere al referente associato alla rivendita.

RV07) Il contatto associato all’ordine dal cliente DEVE appartenere all’insieme dei contatti posseduti dallo stesso.

RV08) Una specie di piante fornita da un fornitore DEVE appartenere alla lista di quelle possedute

## Dizionario dei dati

|  |  |  |  |
| --- | --- | --- | --- |
| **Entità** | **Descrizione** | **Attributi** | **Identificatori** |
| Specie di pianta | Classe d’appartenenza delle piante | Codice, Nome latino, Nome comune | Codice |
| Specie da interni | Particolare sottoinsieme delle specie esistenti | Codice, Nome latino, Nome comune | Codice |
| Specie da esterni | Particolare sottoinsieme delle specie esistenti | Codice, Nome latino, Nome comune | Codice |
| Specie esotica | Particolare sottoinsieme delle specie esistenti | Codice, Nome latino, Nome comune | Codice |
| Specie verde | Particolare sottoinsieme delle specie esistenti | Codice, Nome latino, Nome comune | Codice |
| Specie fiorita | Particolare sottoinsieme delle specie esistenti | Codice, Nome latino, Nome comune, Colorazione | Codice |
| Giacenza | Rappresentazione dello stato del magazzino relativamente ad una data specie di piante | Quantità | Codice (ext. id.) |
| Prezzo | Valore economico di una specie di piante vendute dalla Verde S.r.l. | Valore, Data adozione, Data termine | Codice (ext. id.) + Data adozione |
| Fornitore | Ente che fornisce piante alla Verde S.r.l. | Codice fornitore, Codice fiscale, Nome, Indirizzo | Codice fornitore *or* Codice fiscale |
| Richiesta di Fornitura | Lotto di piante richieste consegnato da un certo fornitore | Quantità,  Data | Quantità +  Codice fornitore (ext. id.) + Codice (ext. id.) |
| Cliente | Acquirente delle piante vendute dalla Verde S.r.l. | Codice, Nome, Indirizzo fatturazione, Indirizzo residenza | Codice |
| Privato | Possibile categoria di clienti gestita da Verde S.r.l. | Codice, Nome, Indirizzo fatturazione, Indirizzo residenza, Codice fiscale | Codice *or* Codice fiscale |
| Rivendita | Possibile categoria di clienti gestita da Verde S.r.l. | Codice, Nome, Indirizzo fatturazione, Indirizzo residenza, Partita IVA | Codice *or* Partita IVA |
| Contatto | Mezzo di comunicazione che permette di rintracciare un cliente e/o un referente della Verde S.r.l. | Valore,  Tipo | Valore |
| Referente | Persona di riferimento per una rivendita cliente della Verde S.r.l. | Nome,  Cognome | Codice cliente (ext. id.) |
| Ordine | Acquisto effettuato da un cliente della Verde S.r.l. | ID, Data, Indirizzo spedizione | ID |
| Ordine aperto | Ordine aperto dal cliente, il quale ancora non ha concluso la fase di inserimento prodotti. | ID, Data, Indirizzo spedizione | ID |
| Ordine finalizzato | Ordine tale per cui l’utente ha terminato l’inserimento di prodotti e che può essere processato dagli operatori. | ID, Data, Indirizzo spedizione | ID |
| Pacco | Unità di partizionamento di un ordine per la spedizione | Numero | Codice (ext. id.) + Data (ext. id.) + Numero |

# Progettazione logica

## Volume dei dati

|  |  |  |
| --- | --- | --- |
| **Concetto nello schema** | **Tipo[[1]](#footnote-1)** | **Volume atteso** |
| Specie di pianta | E |  |
| Specie da interni | E |  |
| Specie da esterni | E |  |
| Specie esotica | E |  |
| Specie verde | E |  |
| Specie fiorita | E |  |
| Giacenza | E |  |
| Magazzino | R |  |
| Prezzo | E |  |
| Adozione | R |  |
| Fornitore | E |  |
| Richiesta di fornitura | E |  |
| Specie richiesta | R |  |
| Fornitore scelto | R |  |
| Disponibilità fornitura | R |  |
| Cliente | E |  |
| Privato | E |  |
| Rivendita | E |  |
| Referente | E |  |
| Contatto | E |  |
| Possesso cliente | R |  |
| Preferenza cliente | R |  |
| Referenza | R |  |
| Possesso referente | R |  |
| Preferenza referente | R |  |
| Ordine | E |  |
| Ordine finalizzato | E |  |
| Ordine aperto | E |  |
| Riferimento | R |  |
| Pacco | E |  |
| Appartenenza | R |  |
| Richiesta ordine | R |  |
| Contatto ordine | R |  |
| Suddivisione | R |  |
| Contenuto | R |  |

## Tavola delle operazioni

|  |  |  |
| --- | --- | --- |
| **Cod.** | **Descrizione** | **Frequenza attesa** |
| XCS-01 | Permette, di aprire un nuovo ordine noti il codice del cliente, l’indirizzo, il contatto e il numero di piante della specie da inserire. |  |
| XCS-02 | Permette di aggiungere una nuova specie ad un ordine esistente noti ambedue gli identificativi. |  |
| XCS-03 | Permette di rimuovere una specie da un ordine esistente noti ambedue gli identificativi. |  |
| XCS-04 | Permette di modificare la quantità di piante di una specie appartenente ad un ordine esistente noti ambedue gli identificativi. |  |
| XCS-05 | Permette di dichiarare finalizzato un ordine di cui si conosce l’identificativo |  |
| XCS-06 | Permette la visualizzazione di un report dettagliato relativo ad un ordine di cui si conosce l’identificativo. |  |
| XCS-07 | Permette di modificare l’indirizzo di residenza di un cliente noto il suo identificativo. |  |
| XCS-08 | Permette di modificare l’indirizzo di fatturazione di un cliente noto il suo identificativo. |  |
| XCS-09 | Permette l’aggiunta di un contatto alla lista dei posseduti di un cliente di cui si conosce il codice. |  |
| XCS-10 | Permette la rimozione di un contatto dalla lista dei posseduti di un cliente di cui si conosce il codice. |  |
| XCS-11 | Permette l’aggiornamento del contattto preferito di un cliente di cui si conosce il codice. |  |
| XCS-12 | Permette di visualizzare l’insieme dei contatti di un cliente di cui si conosce il codice |  |
| XCS-13 | Permette di visualizzare l’insieme degli ordini effettuati da un cliente di cui di conosce il codice |  |
| XCS-14 | Permette di visualizzare l’insieme delle specie appartenenti ad un ordine di cui è noto il codice |  |
| XWM-01 | Permette la ricerca di specie di piante per nome |  |
| RCS-01 | Permette l’aggiunta di un contatto alla lista dei posseduti di un referente di cui si conosce l’identificativo. |  |
| RCS-02 | Permette la rimozione di un contatto dalla lista dei posseduti di un referente di cui si conosce l’identificativo. |  |
| RCS-03 | Permette di aggiornare il contattto preferito di un referente di cui si conosce il codice. |  |
| RCS-04 | Permette di visualizzare l’insieme dei contatti di un referente di cui si conosce il codice |  |
| MNG-01 | Permette l’aggiunta di un nuova specie di piante all’insieme di quelle trattate. |  |
| MNG-02 | Permette la rimozione di una specie di piante dall’insieme delle piante trattate. |  |
| MNG-03 | Permette l’aggiunta di una colorazione possibile per una specie fiorita trattata di cui si conosce il codice. |  |
| MNG-04 | Permette la rimozione di una colorazione possibile per una specie fiorita trattata di cui si conosce il codice. |  |
| MNG-05 | Permette la modifica del prezzo attualmente adottato per una specie di pianta di cui si conosce il codice. |  |
| MNG-06 | Permette la visualizzazione di un report sul trend di vendita di una specie di cui è noto il codice. |  |
| MNG-07 | Permette di visualizzare l’insieme delle colorazioni di una specie fiorita di cui si conosce il codice |  |
| EVN-01 | Permette la rimuozione dall’archivio di informazioni relative ad ordini, richieste di fornitura e prezzi passati |  |
| EVN-02 | Permette l’aggiornamento della giacenza al fronte delle richieste di fornitura effettuate |  |
| WHC-01 | Permette di richiedere ad un fornitore, di cui si conosce l’identificativo una certa quantità di piante di una specie di cui è noto il codice. |  |
| WHC-02 | Permette di inserire un nuovo fornitore nel sistema. |  |
| WHC-03 | Permette di aggiungere una specie alla lista delle disponibili di un fornitore di cui si conosce il codice |  |
| WHC-04 | Permette di visualizzare un report che evidenzia le specie che necessitano un rifornimento della giacenza |  |
| WHC-05 | Permette di visualizzare l’elenco delle specie disponibili di un fornitore di cui è noto il codice |  |
| WHC-06 | Permette di aggiungere un indirizzo alla lista degli indirizzi di un fornitore di cui è noto il codice |  |
| WHC-07 | Permette di visualizzare l’elenco dei fornitori nel sistema |  |
| OPC-01 | Permette la visualizzazione dei dettagli sullo smistamento in pacchi di un’ordine finalizzato di cui si conosce il codice. |  |
| OPC-02 | Permette la creazione di un pacco (inserendovi almeno una pianta) relativo ad un ordine di cui si conosce il codice. |  |
| OPC-03 | Permette l’aggiunta di un certo numero di piante afferenti ad una specie ad un pacco di cui si conosce l’identificativo |  |
| OPC-04 | Permette di visualizzare l’elenco delle specie ancora da impacchettare |  |
| OPC-05 | Permette di visualizzare lo stato di un ordine di cui si conosce il codice |  |
| NRG-01 | Permette l’inserimento di un nuovo cliente privato nella base di dati. | (per si assume l’operazione sia eseguiuta un numero infinitesimo di volte rispetto al valore di ) |
| NRG-02 | Permette l’inserimento di un nuovo cliente rivendita nella base di dati. | (per si assume l’operazione sia eseguiuta un numero infinitesimo di volte rispetto al valore di ) |

## Costo delle operazioni

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione: XCS-01** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Cliente | Entità | 1 | L |
| Pref/Poss cliente | Associazione | 1 | L |
| Contatto | Entità | 1 | L |
| Specie di pianta | Entità | 1 | L |
| Magazzino | Associazione | 1 | L |
| Giacenza | Entità | 1 | L |
| Rivendita | Entità | 1 | L |
| Referenza | Associazione | 1 | L |
| Referente | Entità | 1 | L |
| Ordine | Entità | 1 | S |
| Ordine aperto | Entità | 1 | S |
| Appartenenza | Associazione | 1 | S |
| Richiesta ordine | Associazione | 1 | S |
| Riferimento | Associazione | 1 | S |
| Contatto ordine | Associazione | 1 | S |
| Giacenza | Entità | 1 | S |
| Costo effettivo: 23 | | | |

Nel caso di privati il costo totale decrementa di 5 unità per via dell’assenza del referente (assenti righe 7-8-9-14 della tabella).

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazioni: XCS-02, XCS-04** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Specie di pianta | Entità | 1 | L |
| Magazzino | Associazione | 1 | L |
| Giacenza | Entità | 1 | L |
| Ordine | Entità | 1 | L |
| Appartenenza | Associazione | 1 | S |
| Giacenza | Entità | 1 | S |
| Costo effettivo: 8 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione: XCS-05** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Ordine aperto | Entità | 1 | S (rimozione) |
| Ordine finalizzato | Entità | 1 | S |
| Costo effettivo: 4 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione: XCS-07, XCS-08** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Cliente | Entità | 1 | S |
| Costo effettivo: 2 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione: XCS-09** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Cliente | Entità | 1 | L |
| Contatto | Entità | 1 | S |
| Possesso cliente | Associazione | 1 | S |
| Costo effettivo: 5 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione: XCS-11** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Cliente | Entità | 1 | L |
| Contatto | Entità | 1 | L |
| Preferenza cliente | Associazione | 1 | S |
| Costo effettivo: 4 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione: RCS-01** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Referente | Entità | 1 | L |
| Contatto | Entità | 1 | S |
| Possesso referente | Associazione | 1 | S |
| Costo effettivo: 5 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione: RCS-03** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Referente | Entità | 1 | L |
| Contatto | Entità | 1 | L |
| Preferenza referente | Associazione | 1 | S |
| Costo effettivo: 4 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione: MNG-01** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Specie di pianta | Entità | 1 | S |
| Entità figlie di specie di pianta | Entità | 2/3 | S |
| Magazzino | Associazione | 1 | S |
| Giacenza | Entità | 1 | S |
| Prezzo | Entità | 1 | S |
| Adozione | Associazione | 1 | S |
| Costo effettivo: 10/12 | | | |

L’entità *specie di pianta* possiede 5 entità figlie ad essa collegata. Quattro di esse a due a due formano due generalizzazioni totali a cui si aggiunge una 5 entità figlia che è in generalizzazione parziale e sovrapposta con le precedenti. Se ne deduce che ogni qual volta si procede con la scrittura di un nuovo elemento *specie di pianta*, si procede con 2 oppure 3 scritture aggiuntive per le entità figlie.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione: MNG-03** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Specie fiorita | Entità | 1 | S |
| Costo effettivo: 2 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione: MNG-05** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Specie di pianta | Entità | 1 | L |
| Adozione | Associazione | 1 | S |
| Prezzo | Entità | 2 | S |
| Costo effettivo: 7 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione: WHC-01** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Fornitore | Entità | 1 | L |
| Specie di pianta | Entità | 1 | L |
| Richiesta fornitura | Entità | 1 | S |
| Fornitore scelto | Associazione | 1 | S |
| Specie richiesta | Associazione | 1 | S |
| Costo effettivo: 8 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione: WHC-02** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Specie di pianta | Entità | 1 | L |
| Fornitore | Entità | 1 | S |
| Disponibilità fornitura | Associazione | 1 | S |
| Costo effettivo: 5 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione: WHC-03** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Specie di pianta | Entità | 1 | L |
| Fornitore | Entità | 1 | L |
| Disponibilità fornitura | Associazione | 1 | S |
| Costo effettivo: 4 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione: WHC-6** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Fornitore | Entità | 1 | S |
| Costo effettivo: 2 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione: OPC-02** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Specie di pianta | Entità | 1 | L |
| Ordine finalizzato | Entità | 1 | L |
| Pacco | Entita | 1 | S |
| Contenuto | Associazione | 1 | S |
| Suddivisione | Associazione | 1 | S |
| Costo effettivo: 8 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione: OPC-03** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Specie di pianta | Entità | 1 | L |
| Pacco | Entita | 1 | L |
| Contenuto | Associazione | 1 | S |
| Costo effettivo: 4 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione: NRG-01** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Cliente | Entità | 1 | S |
| Cliente privato | Entità | 1 | S |
| Costo effettivo: 4 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operazione: NRG-02** | | | |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo** |
| Cliente | Entità | 1 | S |
| Cliente rivendita | Entità | 1 | S |
| Referenza | Associazione | 1 | S |
| Referente | Entità | 1 | S |
| Costo effettivo: 8 | | | |

## Ristrutturazione dello schema E-R

**Analisi delle ridondanze**

In questa fase di ristrutturazione è stato scelto di eliminare l’associazione *riferimento* che lega *ordine* con *referente.*

Infatti l’entità *referente* è in associazione 1:1 con l’entità *rivendita* che già è associata (tramite l’entità padre) all’*ordine* mediante l’associazione *richiesta d’ordine*. Non essendo presenti operazioni che richiedono esclusivamente l’accesso al referente associato ad un’ordine senza conoscere i dettagli della rivendita corrispondente è preferibile rinunciare alla ridondanza per diminuire il costo dell’operazione XCS-01 interessata.

Operazione XCS-01: COSTO TOTALE: (risparmio di 3 letture ed una scrittura).

È stato inoltre aggiunto l’attributo pendente all’entità *richiesta di fornitura* per rendere più adattabile il sistema a modifichè future. Di fatti in questo scenario sarebbe possibile verificare se un rifornimento è stato realmente effettuato basandosi soltanto sulla data poiché l’operazione EVN-02 settimanalmente aggiorna le giacenze al fronte di richieste.

Se in futuro si implementasse un aggiornamento manuale asincrono delle giacenze si rischierebbe di aggiornare più volte le tuple rendendo lo stato della base di dati inconsistente rispetto al magazzino reale.

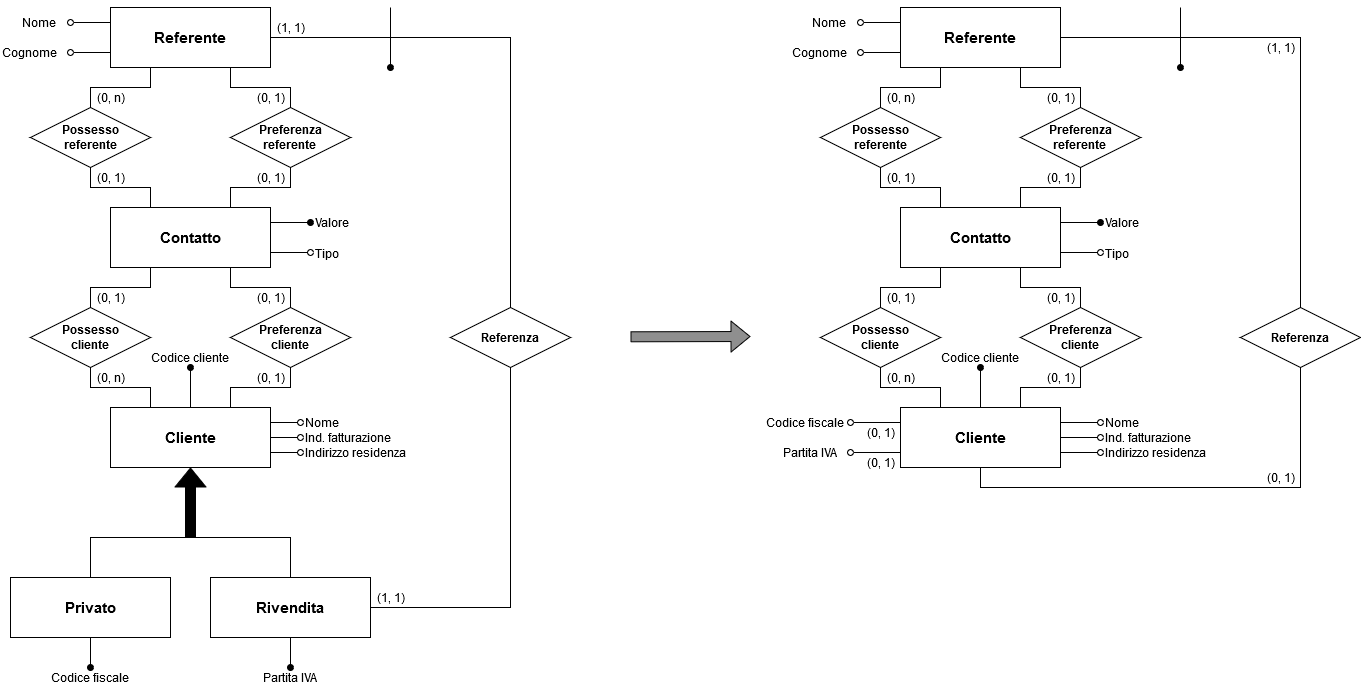
**Eliminazione delle generalizzazioni**

Per tutte e tre le generalizzazioni presenti nello schema E-R è stato scelto in questa fase progettuale l’accorpamento all’entità padre come strategia di eliminazione.

Per la generalizzzazione *cliente-privato-rivendita,* la scelta è legata alla mancanza di distinzione fra le due specializzazioni nelle operazioni elencate (praticamente tutte si riferiscono al cliente in generale) e nelle associazioni presenti nel diagramma E-R (fatta eccezione di *referenza*).

Per questo motivo la cardinalità minima di tale associazione rispetto a *cliente* subisce una variazione da 1 a 0 (per i privati non è richiesto di memorizzare il referente).

Non è stato introdotto l’attributo tipo in questa fase d’analisi poiché gli attributi codice fiscale e partita IVA rispettivamente di privato e rivendita sono mutalmente esclusivi in cliente (l’assenza di uno di essi determina univocamente l’una o l’altra specializzazione).

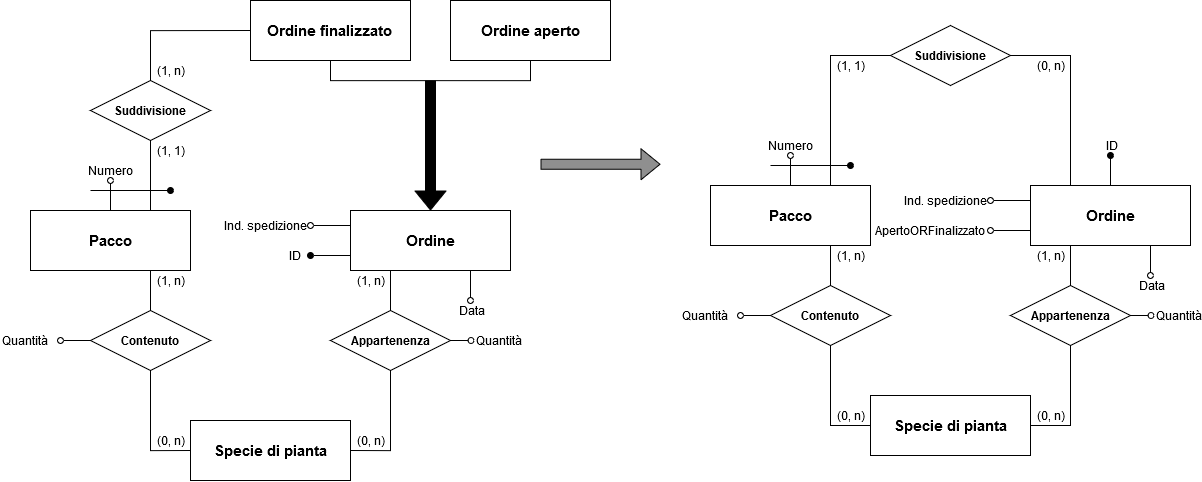


Per la generalizzazione *ordine-ordine finalizzato-ordine aperto* la scelta è legata alla natura stessa della generalizzazione: essa era stata introdotta in fase di progettazione concettuale con l’unico obiettivo di distinguere due diverse tipologie di ordini. Ai fini dell’ottimizzazione in questa fase è stato preferito inserire l’attributo *ApertoORFinalizzato* a tale scopo.

L’operazione XCS-05 è l’unica interessata dalla modifica e, come viene mostrato qui sotto, subisce un notevole miglioramento.

Operazione XCS-05: COSTO TOTALE: (risparmio di una scrittura).

L’associazione *suddivisione* rispetto ad *ordine finalizzato* a seguito della modifica subisce una variazione di cardinalità minima (da (1, n) a (0, n)).



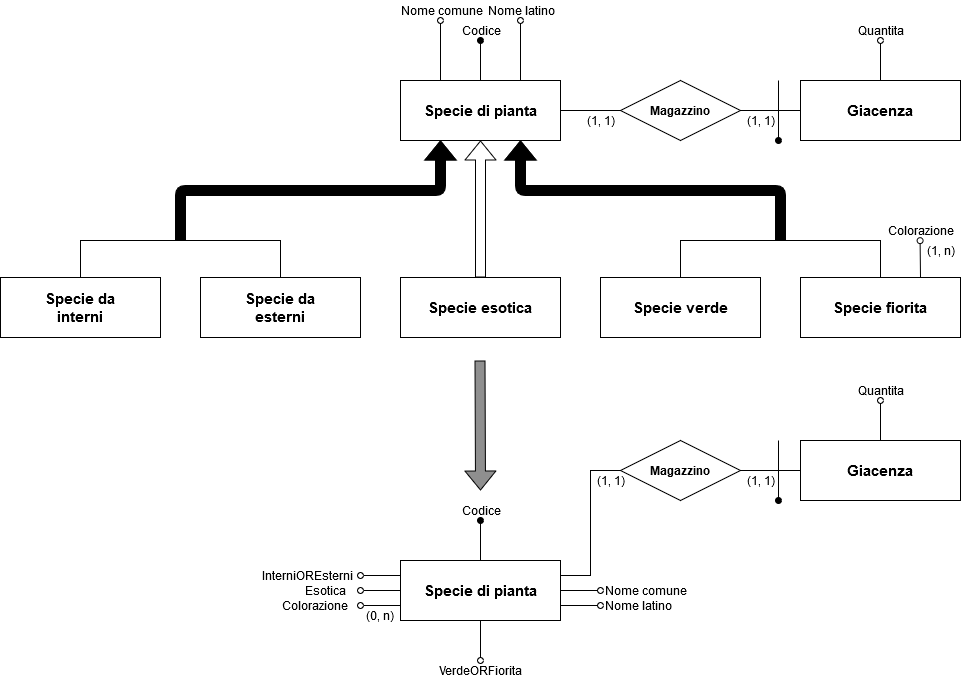
Per la generalizzazione relativa alla *specie di pianta* infine la scelta è legata a motivazioni analoghe al caso precedente.

Anche in questo caso le varie entità figlie non sono altro che diverse categorie di specie che risulta essere l’unica entità interessata da operazioni ed associazioni nel diagramma E-R.

Inoltre tale accorpamento causa una miglioria sul costo dell’operazione MN01:

Operazione MN01: COSTO TOTALE: (risparmio di 2 o 3 scritture).

Per la rimozione della generalizzazione totale che coinvolge *specie da interni* e *specie da esterni* si introduce lì attributo *InterniOREsterni*, per la generalizzazione parziale *specie esotica* si l’attributo *Esotica*, infine per la generalizzazione totale relativa a *specie verde* o *fiorita* l’attributo *VerdeORFiorita.*



**Partizionamento/Accorpamento entità e associazioni**

L’unica operazione di accorpamento effettuata sullo schema prodotto fin qui è quella che coinvolge le entità *specie di pianta* e *giacenza.*

Sebbene le due furono divise in fase di progettazione concettuale per motivi semantici in questa fase, al fine di ottimizzare le prestazioni è stato scelto l’accorpamento.

In quasi ogni operazione che coinvolge la *specie di pianta* è di interesse conoscere la giacenza in magazzino della stessa.

Ciascuna operazione che prevedeva 3 letture/scritture distinte nello schema concettuale originario, in quello di sotto riportato ora ne necessita soltanto 1.

**Immagine che contiene orologio

Descrizione generata automaticamente**

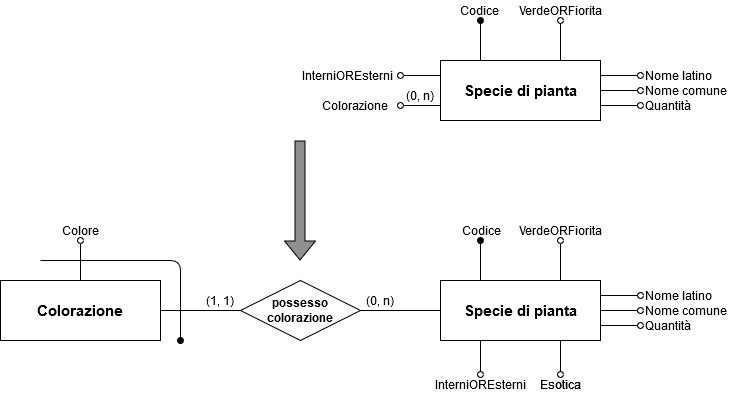
Nella fase di partizionamento di entità invece sono stati reificati gli attributi multivalore presenti nello schema per adattarli al modello relazionale.

L’attributo indirizzo del fornitore (cardinalità (1, n)) dell’entità *fornitore* viene reificato ad entità e viene collegato mediante un’associazione 1:n come viene mostrato nello schema sottostante.

Immagine che contiene orologio

Descrizione generata automaticamente

L’attributo colorazione dell’entità *specie di pianta* (cardinalità (0, n)) viene reificato ad entità e viene collegato mediante un associazione 0:n come viene mostrato nello schema sottostante.



**Scelta degli identificatori primari**

Di seguito vengono riportate specifiche ulteriori sulle scelte degli identificatori. Le entità che non compaiono in questa lista mantengono l’identificatore presentato nel diagramma E-R.

* **Fornitore**

per tale entità si sceglie come identificatore primario il codice fornitore poiché più pertinente al dominio di interesse rispetto al codice fiscale.

* **Cliente**

per tale entità si accorpano gli attributi codice (che era stato inserito in fase progettuale senza dirette indicazioni provenienti dalla descrizione del minimondo) partita IVA e codice fiscale in una nuova versione dell’attributo codice*.* Viene in aggiunta introdotto un attributo *PrivatoORRivendita* per discriminare la categoria di cliente.

In questo modo si passa da una configurazione con tre attributi di cui due opzionali, ad un'altra con soli 2 attributi non opzionali.

Il nuovo identificatore corrisponderà al valore del codice fiscale nel caso di un privato o alla partita iva nel caso di una rivendita.

ESEMPIO:

1. elementi appartenenti all’entità cliente prima del cambio di identifiatore:

c1(00001, “Mario”, “Via Roma, 29”, “Via Roma, 29”, “RSSMRA80A01H501U”, null)

c2(00002, “Flora”, “Via Verdi, 1”, “Via Torino, 12”, null, “0764352056C”)

1. elemento appartenente all’entità cliente dopo il cambio di identificatore:

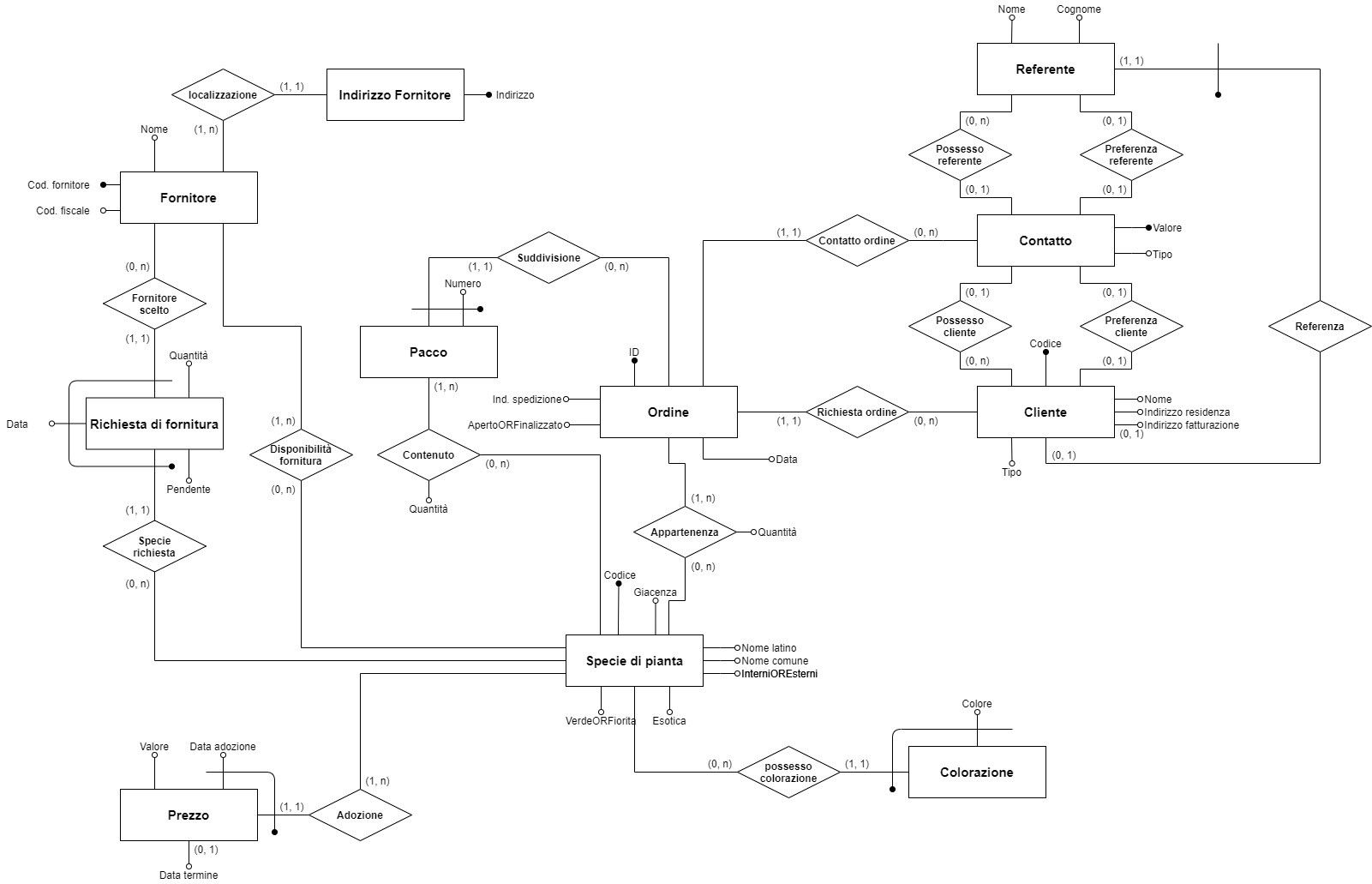
c1(“RSSMRA80A01H501U”, “Mario”, “Via Roma, 29”, “Via Roma, 29”, P)

c2(“0764352056C”, “Flora”, “Via Verdi, 1”, “Via Torino, 12”, R)

## Trasformazione di attributi e identificatori

Non vi sono attributi ripetuti e tutti gli identificatori esterni finora mantenuti rimangono inalterati.

## Traduzione di entità e associazioni



La convenzione adottata nella traduzione è stata quella di utilizzare underscore e caratteri minuscoli non accentati per rappresentare sia relazioni che attributi, ed inoltre l’adozione del plurale per i nomi delle relazioni anziché il singolare utilizzato nella nomenclatura delle entità modello concettuale.

Inoltre per una maggiore chiarezza nella traduzione sono state attuate le seguente modifiche ai nomi:

* le associazioni *possesso cliente/referente* sono state tradotte in *contatti\_cliente/referente*
* gli attributi esotica dell’entità *specie di pianta* e pendente dell’entità *richiesta di fornitura* sono stati tradotti rispettivamente in esotica\_si\_no e pendente\_si\_no *.*
* le associazioni *appartenenza* e *contenuto* sono state tradotte rispettivamente in appartenenza\_ordini e contenuto\_pacchi.

La traduzione nel modello relazionale porta al seguente schema:

**appartenenza\_ordini** (ordine, specie\_di\_pianta, quantita)

**clienti** (codice, nome, indirizzo\_residenza, indirizzo\_fatturazione\*, privato\_or\_rivendita, contatto\_preferito\*)

**colorazioni** (specie\_fiorita, colore)

**contatti** (valore, tipo)

**contatti\_clienti** (contatto, cliente)

**contatti\_referenti** (contatto, referente)

**contenuto\_pacchi** (ordine, pacco, specie\_di\_pianta, quantita)

**disponibilita\_fornitura** (specie\_di\_pianta , fornitore)

**fornitori** (codice\_fornitore, codice\_fiscale, nome)

**indirizzi\_fornitori** (indirizzo, fornitore)

**ordini** (id, cliente, data, aperto\_or\_finalizzato, indirizzo\_spedizione, contatto)

**pacchi** (ordine, numero)

**prezzi** (specie\_di\_pianta, data\_adozione, valore, data\_termine\*)

**referenti** (rivendita, nome, cognome, contatto\_preferito\*)

**richieste\_di\_forniture** (data, fornitore\_scelto, specie\_richiesta, quantita, pendente\_si\_no)

**specie\_di\_piante** (codice, nome\_latino, nome\_comune, interni\_or\_esterni, verde\_or\_fiorita, esotica\_si\_no, giacenza)

Con i seguenti vincoli di integrita refernziale:

**appartenenza\_ordini** (ordine) **ordini** (id)

**appartennza\_ordini** (specie\_di\_pianta) **specie\_di\_piante** (codice)

**clienti** (contatto\_preferito) **contatti** (valore)

**colorazioni** (specie\_fiorita) **specie\_di\_piante** (codice)

**contatti\_clienti** (contatto) **contatti** (valore)

**contatti\_clienti** (cliente) **clienti** (codice)

**contatti\_referenti** (contatto) **contatti** (valore)

**contatti\_referenti** (cliente) **clienti** (codice)

**contenuto\_pacchi** (ordine, pacco) **pacchi** (ordine, numero)

**contenuto\_pacchi** (specie\_di\_pianta) **specie\_di\_piante** (codice)

**disponibilita\_forniture** (specie\_di\_pianta) **specie\_di\_piante** (codice)

**disponibilita\_forniture** (fornitore) **fornitori** (codice\_fornitore)

**indirizzi\_fornitori** (fornitore) **fornitori** (codice\_fornitore)

**ordini** (clienti) **clienti** (codice)

**ordini** (contatto) **contatti** (valore)

**pacchi** (ordine) **ordini** (id)

**prezzi** (specie\_di\_pianta) **specie\_di\_piante** (codice)

**referenti** (rivendita) **clienti** (codice)

**referenti** (contatto\_preferito) **contatti** (valore)

**richieste\_di\_forniture** (fornitore\_scelto) **fornitori** (codice\_fornitore)

**richieste\_di\_forniture** (pianta\_richiesta) **specie\_di\_piante** (codice)

## Normalizzazione del modello relazionale

Per dimostrare che lo schema prodotto soddisfa le forme 1NF, 2NF e 3NF è sufficiente mostrare come soddisfi la BCNF.

Per provare tale proprietà è necessario mostrare che per ogni dipendenza funzionale del tipo X Y, X è superchiave per la relazione.

Per la maggior parte delle relazioni, avendo esse pochi attributi, è facile dimostrarlo.

Esempio: relazione **indirizzi fornitori**

indirizzo fornitore, ma fornitore indirizzo poiché ad esempio un fornitore può avere due indirizzi differenti

Per quanto riguarda le relazioni **specie\_di\_piante** e **fornitori** il discorso si amplia anche alla superchiave essendo presenti più chiavi candidate

Esempio: relazione **specie\_di\_piante**

codice (nome\_latino, nome\_comune, … esotica\_si\_no, giacenza)

nome\_latino (codice, nome\_comune, … esotica\_si\_no, giacenza)

(non esistono campi Y tali per cui codice Y ma nome\_latino Y)

Estendendo il ragionamento alle altre relazioni presenti nello schema è possibile concludere che lo schema è in BCNF.

# Progettazione fisica

## Utenti e privilegi

Dall’analisi della specifica del minimondo si ricavano le seguenti classsi d’utenti:

1. Cliente
   1. Cliente rivendita [RCS]
   2. Cliente privato [PCS]
2. Manager [MNG]
3. Operatore pacchi [OPC]
4. Adetto dipartimento magazzino [WHC]

Alle quali si aggiungono due ulteriori classi d’utenza necessarie per l’ingegnerizzazione del sistema:

1. Capo del personale [COS]
2. Non registrato [NRG]

Prima di effettuare una digressione sui privilegi delle classi d’utenza sulla base di dati è necessario introdurre due nuove relazioni:

**utenti** (username, password, ruolo, uuid)

**utenze\_clienti** (cliente, utente)

con i seguenti vincoli di integrita referenziale:

**utenze\_clienti** (cliente) **clienti** (codice)

**utenze\_clienti** (utente) **utenti** (username)

Lo scopo della prima è quello di memorizzare le credenziali di accesso e la classe di appartenenza dei vari utenti fisici che si connettono al database. La seconda è invece la tabella pivot di una associazione 1:1 opzionale su cui si è adottata una traduzione non ottima.

Il suo scopo è quello di associare un utenza di un cliente alle sue informazioni effettive memorizzate nella base di dati.

In questa fase vengono introdotte anche altre 3 nuove operazioni:

1. NRG-03: permette di effettuare l’accesso al sistema
2. ALL-01: permette la modifica della password d’accesso
3. COS-01: permette di creare un’utenza dipendente (classi MNG, WHC, OPC, COS)

Anziché assegnare privilegi di lettura e/o scritture ad intere tabelle è stato preferito conferire il solo privilegio di esecuzione di alcune stored procedure a ciascuna classe d’utenza.

Nella tabella seguente vengono riassunti i privilegi d’esecuzione di ciascuna classe d’utenza presentata:

|  |  |
| --- | --- |
| **Classe d’utenza** | **Privilegi (EXECUTE)** |
| Cliente rivendita | XCS-$$  RCS-$$  ALL-01 |
| Cliente privato | XCS-$$  ALL-01 |
| Manager | MNG-$$  ALL-01 |
| Operatore pacchi | OPC-$$  ALL-01 |
| Addetto dipartimento magazzino | WHC-$$  ALL-01 |
| Capo del personale | COS-01  ALL-01 |
| Non registrato | NRG-$$ |

**n.b.** I caratteri speciali $$ corrispondono ai numeri 01, 02 … presentati nella sezione relativa alle operazioni.

## Strutture di memorizzazione

|  |  |  |
| --- | --- | --- |
| **Tabella apparteneza\_ordini** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[2]](#footnote-2)** |
| ordine | INT | PK. NN |
| specie\_di\_pianta | INT | PK, NN |
| quantita | INT | NN, UN |

|  |  |  |
| --- | --- | --- |
| **Tabella clienti** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| codice | VARCHAR(16) | PK. NN |
| nome | VARCHAR(32) | NN |
| indirizzo\_residenza | VARCHAR(64) | NN |
| indirizzo\_fatturazione | VARCHAR(64) |  |
| privato\_or\_rivendita | TINYINT | NN |
| contatto\_preferito | VARCHAR(256) |  |

|  |  |  |
| --- | --- | --- |
| **Tabella colorazioni** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| specie\_fiorita | INT | PK. NN |
| colore | VARCHAR(32) | PK, NN |

|  |  |  |
| --- | --- | --- |
| **Tabella contatti** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| valore | VARCHAR(256) | PK. NN |
| tipo | TINYINT | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella contatti\_clienti** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| contatto | VARCHAR(256) | PK, NN |
| cliente | VARCHAR(16) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella contatti\_referenti** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| contatto | VARCHAR(256) | PK. NN |
| referente | VARCHAR(16) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella contenuto\_pacchi** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| ordine | INT | PK. NN |
| pacco | INT | PK, NN, UN |
| specie\_di\_pianta | INT | PK, NN |
| quantita | INT | NN, UN |

|  |  |  |
| --- | --- | --- |
| **Tabella disponibilita\_forniture** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| specie\_di\_pianta | INT | PK. NN |
| fornitore | INT | PK, NN |

|  |  |  |
| --- | --- | --- |
| **Tabella fornitori** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| codice\_fornitore | INT | PK. NN, AI |
| codice\_fiscale | VARCHAR(16) | NN, UQ |
| nome | VARCHAR(32) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella indirizzi\_fornitori** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| indirizzo | VARCHAR(64) | PK, NN |
| fornitore | INT | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella ordini** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| id | INT | PK. NN, AI |
| cliente | VARCHAR(16) | NN |
| data | DATETIME | NN |
| aperto\_or\_finalizzato | TINYINT | NN |
| indirizzo\_spedizione | VARCHAR(64) | NN |
| contatto | VARCHAR(256) | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella pacchi** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| ordine | INT | PK. NN |
| numero | INT | PK, NN, UN |

|  |  |  |
| --- | --- | --- |
| **Tabella prezzi** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| specie\_di\_pianta | INT | PK. NN |
| data\_adozione | DATETIME | PK, NN |
| valore | DECIMAL(7,2) | NN, UN |
| data\_termine | DATETIME |  |

|  |  |  |
| --- | --- | --- |
| **Tabella referenti** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| rivendita | VARCHAR(16) | PK, NN |
| nome | VARCHAR(32) | NN |
| cognome | VARCHAR(32) | NN |
| contatto\_preferito | VARCHAR(256) |  |

|  |  |  |
| --- | --- | --- |
| **Tabella richieste\_di\_forniture** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| data | DATETIME | PK. NN |
| fornitore\_scelto | INT | PK, NN |
| specie\_richiesta | INT | PK, NN |
| quantita | INT | NN, UN |
| pendente\_si\_no | TINYINT | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella specie\_di\_piante** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| codice | INT | PK. NN, AI |
| nome\_latino | VARCHAR(64) | NN, UQ |
| nome\_comune | VARCHAR(64) | NN |
| interni\_or\_esterni | TINYINT | NN |
| verde\_or\_fiorita | TINYINT | NN |
| esotica\_si\_no | TINYINT | NN |
| giacenza | INT | NN, UN |

|  |  |  |
| --- | --- | --- |
| **Tabella utenti** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| username | VARCHAR(128) | PK, NN |
| password | CHAR(128) | NN |
| ruolo | ENUM | NN |
| uuid | CHAR(36) | NN, UQ |

|  |  |  |
| --- | --- | --- |
| **Tabella utenze\_clienti** | | |
| **Attributo** | **Tipo di dato** | **Attributi** |
| cliente | VARCHAR(16) | PK, NN |
| utente | VARCHAR(32) | NN |

## Indici

L’unico indice inserito manualmente in fase di modellazione è il seguente:

|  |  |
| --- | --- |
| **Tabella specie\_di\_piante** | |
| **Indice nome\_comune\_FULLTEXT** | **Tipo[[3]](#footnote-3):** |
| nome\_comune 1 | FT |

Tale indice è stato creato per velocizzare la ricerca nell’operazione XWM-01 ed inoltre è stato scelto il tipo FULLTEXT poiché tale si basa sull’utilizzo dell’operatore LIKE.

I seguenti indici sono invece stati autogenerati:

|  |  |
| --- | --- |
| **Tabella appartenenza\_ordini** | |
| **Indice PRIMARY** | **Tipo:** |
| ordine 1 specie\_di\_pianta 2 | PRIMARY |
| **Indice fk\_appartenenza\_ordini\_specie\_di\_piante1\_idx** | **Tipo:** |
| specie\_di\_pianta 1 | IDX |
| **Indice fk\_appartenenza\_ordini\_ordini1\_idx** | **Tipo:** |
| ordine 1 | IDX |

|  |  |
| --- | --- |
| **Tabella clienti** | |
| **Indice PRIMARY** | **Tipo:** |
| codice 1 | PRIMARY |
| **Indice fk\_clienti\_contatti1\_idx** | **Tipo:** |
| contatto\_preferito 1 | IDX |

|  |  |
| --- | --- |
| **Tabella colorazioni** | |
| **Indice PRIMARY** | **Tipo:** |
| specie\_fiorita 1 colore 2 | PRIMARY |

|  |  |
| --- | --- |
| **Tabella contatti** | |
| **Indice PRIMARY** | **Tipo:** |
| valore 1 | PRIMARY |

|  |  |
| --- | --- |
| **Tabella contatti\_clienti** | |
| **Indice PRIMARY** | **Tipo:** |
| contatto 1 | PRIMARY |
| **Indice fk\_contatti\_clienti\_clienti1\_idx** | **Tipo:** |
| cliente 1 | IDX |

|  |  |
| --- | --- |
| **Tabella contatti\_referenti** | |
| **Indice PRIMARY** | **Tipo:** |
| contatto 1 | PRIMARY |
| **Indice fk\_contatti\_referenti\_referenti1\_idx** | **Tipo:** |
| referente 1 | IDX |

|  |  |
| --- | --- |
| **Tabella contenuto\_pacchi** | |
| **Indice PRIMARY** | **Tipo:** |
| ordine 1 pacco 2 specie\_di\_pianta 3 | PRIMARY |
| **Indice fk\_contenuto\_pacchi\_specie\_di\_piante1\_idx** | **Tipo:** |
| specie\_di\_pianta 1 | IDX |
| **Indice fk\_contenuto\_pacchi\_pacchi1\_idx** | **Tipo:** |
| ordine 1 pacco 2 | IDX |

|  |  |
| --- | --- |
| **Tabella disponibilita\_forniture** | |
| **Indice PRIMARY** | **Tipo:** |
| specie\_di\_pianta 1 fornitore 2 | PRIMARY |
| **Indice fk\_disponibilita\_forniture\_specie\_di\_piante1\_idx** | **Tipo:** |
| specie\_di\_pianta 1 | IDX |
| **Indice fk\_ disponibilita\_forniture\_fornitori1\_idx** | **Tipo:** |
| fornitore 1 | IDX |

|  |  |
| --- | --- |
| **Tabella fornitori** | |
| **Indice PRIMARY** | **Tipo:** |
| codice\_fornitore 1 | PRIMARY |
| **Indice codice\_fiscale\_UNIQUE** | **Tipo:** |
| codice\_fiscale 1 | UQ |

|  |  |
| --- | --- |
| **Tabella indirizzi\_fornitori** | |
| **Indice PRIMARY** | **Tipo:** |
| indirizzo 1 | PRIMARY |
| **Indice fk\_indirizzi\_fornitori\_fornitori1\_idx** | **Tipo:** |
| fornitore 1 | IDX |

|  |  |
| --- | --- |
| **Tabella ordini** | |
| **Indice PRIMARY** | **Tipo:** |
| id 1 | PRIMARY |
| **Indice fk\_ordini\_contatti1\_idx** | **Tipo:** |
| contatto 1 | IDX |
| **Indice fk\_ ordini\_clienti1\_idx** | **Tipo:** |
| cliente 1 | IDX |

|  |  |
| --- | --- |
| **Tabella pacchi** | |
| **Indice PRIMARY** | **Tipo:** |
| ordine 1 numero 2 | PRIMARY |
| **Indice fk\_pacchi\_ordini1\_idx** | **Tipo:** |
| ordine 1 | IDX |

|  |  |
| --- | --- |
| **Tabella prezzi** | |
| **Indice PRIMARY** | **Tipo:** |
| specie\_di\_pianta 1 data\_adozione 2 | PRIMARY |
| **Indice fk\_prezzi\_specie\_di\_piante1\_idx** | **Tipo:** |
| specie\_di\_pianta 1 | IDX |

|  |  |
| --- | --- |
| **Tabella referenti** | |
| **Indice PRIMARY** | **Tipo:** |
| rivendita 1 | PRIMARY |
| **Indice fk\_referenti\_contatti1\_idx** | **Tipo:** |
| contatto\_preferito 1 | IDX |

|  |  |
| --- | --- |
| **Tabella richieste\_di\_forniture** | |
| **Indice PRIMARY** | **Tipo:** |
| data 1 fornitore\_scelto 2 specie\_richiesta 3 | PRIMARY |
| **Indice fk\_richieste\_di\_forniture\_specie\_di\_piante1\_idx** | **Tipo:** |
| specie\_richiesta 1 | IDX |
| **Indice fk\_ richieste\_di\_forniture\_fornitori1\_idx** | **Tipo:** |
| fornitore\_scelto 1 | IDX |

|  |  |
| --- | --- |
| **Tabella referenti** | |
| **Indice PRIMARY** | **Tipo:** |
| rivendita 1 | PRIMARY |
| **Indice fk\_referenti\_contatti1\_idx** | **Tipo:** |
| contatto\_preferito 1 | IDX |

|  |  |
| --- | --- |
| **Tabella specie\_di\_piante** | |
| **Indice PRIMARY** | **Tipo:** |
| codice 1 | PRIMARY |
| **Indice nome\_latino\_UNIQUE** | **Tipo:** |
| nome\_latino 1 | UQ |

|  |  |
| --- | --- |
| **Tabella utenti** | |
| **Indice PRIMARY** | **Tipo:** |
| username 1 | PRIMARY |
| **Indice uuid\_UNIQUE** | **Tipo:** |
| uuid 1 | UQ |

|  |  |
| --- | --- |
| **Tabella utenze\_clienti** | |
| **Indice PRIMARY** | **Tipo:** |
| cliente 1 | PRIMARY |
| **Indice fk\_utenze\_clienti\_utenti1\_idx** | **Tipo:** |
| utente 1 | IDX |

Il DBMS ha quindi autogenerato:

1. degli indici di tipo PRIMARY per ciascuna chiave primaria
2. degli indici di tipo INDEX per ciascuna foreign key per velocizzare le operazioni di join.
3. degli indici di tipo UNIQUE per ciascun campo con attributo UNIQUE presente nello schema, anche in questo caso per facilitare le operazioni di ricerca.

## Trigger

**Tabella utenti**

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`utenti\_BEFORE\_INSERT`

BEFORE INSERT ON `utenti` FOR EACH ROW

BEGIN

IF INSTR(NEW.`username`, " ") > 0 THEN

SIGNAL SQLSTATE '45004'

SET MESSAGE\_TEXT = "Username must not contains spaces";

END IF;

END$$

**Tabella specie\_di\_piante**

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`specie\_di\_piante\_BEFORE\_INSERT`

BEFORE INSERT ON `specie\_di\_piante` FOR EACH ROW

BEGIN

IF NEW.interni\_or\_esterni <> 1 AND NEW.interni\_or\_esterni <> 0 THEN

SIGNAL SQLSTATE '45022'

SET MESSAGE\_TEXT = "Types of species allowed: 1 -> Indoor ~ 0 -> Outdoor";

END IF;

IF NEW.verde\_or\_fiorita <> 1 AND NEW.verde\_or\_fiorita <> 0 THEN

SIGNAL SQLSTATE '45022'

SET MESSAGE\_TEXT = "Types of species allowed: 1 -> Green ~ 0 -> Flowery";

END IF;

IF NEW.esotica\_si\_no <> 1 AND NEW.esotica\_si\_no <> 0 THEN

SIGNAL SQLSTATE '45022'

SET MESSAGE\_TEXT = "Types of species allowed: 1 -> Exotic ~ 0 -> Not exotic";

END IF;

END$$

**Tabella clienti**

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`clienti\_BEFORE\_INSERT`

BEFORE INSERT ON `clienti` FOR EACH ROW

BEGIN

IF NEW.codice REGEXP "[A-Za-z]{6}[0-9]{2}[A-Za-z]{1}[0-9]{2}[A-Za-z]{1}[0-9]{3}[A-Za-z]{1}$" THEN

IF NEW.privato\_or\_rivendita <> 1 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = "Mismatch between code type and customer type";

END IF;

ELSEIF NEW.codice REGEXP "[0-9]{11}$" THEN

IF NEW.privato\_or\_rivendita <> 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = "Mismatch between code type and customer type";

END IF;

ELSE

SIGNAL SQLSTATE '45001' SET MESSAGE\_TEXT = "Wrong code inserted";

END IF;

END$$

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`clienti\_BEFORE\_UPDATE`

BEFORE UPDATE ON `clienti` FOR EACH ROW

BEGIN

DECLARE var\_proprietario VARCHAR(16);

IF NEW.contatto\_preferito IS NOT NULL AND

NEW.contatto\_preferito <> OLD.contatto\_preferito THEN

SELECT cliente

FROM contatti\_clienti

WHERE contatto = NEW.contatto\_preferito

INTO var\_proprietario;

IF NEW.codice <> var\_proprietario THEN

SIGNAL SQLSTATE '45009'

SET MESSAGE\_TEXT = "Selected customer does not own this contact";

END IF;

END IF;

END$$

**Tabella referenti**

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`referenti\_BEFORE\_UPDATE`

BEFORE UPDATE ON `referenti` FOR EACH ROW

BEGIN

DECLARE var\_proprietario VARCHAR(16);

IF NEW.contatto\_preferito IS NOT NULL AND

NEW.contatto\_preferito <> OLD.contatto\_preferito THEN

SELECT referente

FROM contatti\_referenti

WHERE contatto = NEW.contatto\_preferito

INTO var\_proprietario;

IF NEW.rivendita <> var\_proprietario THEN

SIGNAL SQLSTATE '45009'

SET MESSAGE\_TEXT = "Selected referent does not own this contact";

END IF;

END IF;

END$$

**Tabella pacchi**

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`pacchi\_BEFORE\_INSERT`

BEFORE INSERT ON `pacchi` FOR EACH ROW

BEGIN

DECLARE var\_stato TINYINT;

SELECT aperto\_or\_finalizzato

FROM ordini

WHERE id = NEW.ordine

INTO var\_stato;

IF var\_stato = 1 THEN

SIGNAL SQLSTATE '45034' SET MESSAGE\_TEXT = "Order not yet finalized";

END IF;

END$$

**Tabella fornitori**

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`fornitori\_BEFORE\_INSERT`

BEFORE INSERT ON `fornitori` FOR EACH ROW

BEGIN

IF NEW.codice\_fiscale REGEXP "/^[A-Za-z]{6}[0-9]{2}[A-Za-z]{1}[0-9]{2}[A-Za-z]{1}[0-9]{3}[A-Za-z]{1}$/" THEN

SIGNAL SQLSTATE '45023'

SET MESSAGE\_TEXT = "Fiscal code expression not valid";

END IF;

END$$

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`fornitori\_BEFORE\_INSERT\_1`

BEFORE INSERT ON `fornitori` FOR EACH ROW

BEGIN

SET NEW.codice\_fiscale = UPPER(NEW.codice\_fiscale);

END$$

**Tabella richieste\_di\_forniture**

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`richieste\_di\_forniture\_BEFORE\_INSERT`

BEFORE INSERT ON `richieste\_di\_forniture` FOR EACH ROW

BEGIN

DECLARE var\_verifica\_disponibilita INT;

SELECT COUNT(\*)

FROM disponibilita\_forniture

WHERE fornitore = NEW.fornitore\_scelto AND

specie\_di\_pianta = NEW.specie\_richiesta

INTO var\_verifica\_disponibilita;

IF var\_verifica\_disponibilita = 0 THEN

SIGNAL SQLSTATE '45024'

SET MESSAGE\_TEXT = "Request species not available";

END IF;

END$$

**Tabella colorazioni**

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`colorazioni\_BEFORE\_INSERT`

BEFORE INSERT ON `colorazioni` FOR EACH ROW

BEGIN

DECLARE var\_verde\_or\_fiorita TINYINT;

SELECT verde\_or\_fiorita

FROM specie\_di\_piante

WHERE codice = NEW.specie\_fiorita

INTO var\_verde\_or\_fiorita;

IF var\_verde\_or\_fiorita = 1 THEN

SIGNAL SQLSTATE '45019'

SET MESSAGE\_TEXT = "Green species cannot be colorful";

END IF;

END$

**Tabella contenuto\_pacchi**

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`contenuto\_pacchi\_BEFORE\_INSERT`

BEFORE INSERT ON `contenuto\_pacchi` FOR EACH ROW

BEGIN

DECLARE var\_verifica\_appartenenza INT;

SELECT COUNT(\*)

FROM appartenenza\_ordini

WHERE ordine = NEW.ordine AND

specie\_di\_pianta = NEW.specie\_di\_pianta

INTO var\_verifica\_appartenenza;

IF var\_verifica\_appartenenza = 0 THEN

SIGNAL SQLSTATE '45020'

SET MESSAGE\_TEXT = "Species does not belong to the order";

END IF;

END$$

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`contenuto\_pacchi\_BEFORE\_INSERT\_1`

BEFORE INSERT ON `contenuto\_pacchi` FOR EACH ROW

BEGIN

DECLARE var\_quantita\_ordine INT;

DECLARE var\_gia\_impacchettate INT;

SELECT quantita

FROM appartenenza\_ordini

WHERE specie\_di\_pianta = NEW.specie\_di\_pianta AND

ordine = NEW.ordine

INTO var\_quantita\_ordine;

SELECT SUM(quantita)

FROM contenuto\_pacchi

WHERE ordine = NEW.ordine AND

specie\_di\_pianta = NEW.specie\_di\_pianta

INTO var\_gia\_impacchettate;

IF var\_gia\_impacchettate IS NULL THEN

SET var\_gia\_impacchettate = 0;

END IF;

IF NEW.quantita > var\_quantita\_ordine - var\_gia\_impacchettate THEN

SIGNAL SQLSTATE '45015'

SET MESSAGE\_TEXT = "Selected quantity greater than that not processed yet";

END IF;

END$$

**Tabella appartenenza\_ordini**

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`appartenenza\_ordini\_BEFORE\_INSERT`

BEFORE INSERT ON `appartenenza\_ordini` FOR EACH ROW

BEGIN

DECLARE var\_giacenza INT UNSIGNED;

SELECT giacenza

FROM specie\_di\_piante

WHERE codice = NEW.specie\_di\_pianta

INTO var\_giacenza;

IF NEW.quantita > var\_giacenza THEN

SIGNAL SQLSTATE '45006'

SET MESSAGE\_TEXT = "Selected quantity is higher than current stock";

END IF;

END$$

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`appartenenza\_ordini\_BEFORE\_INSERT\_1`

BEFORE INSERT ON `appartenenza\_ordini` FOR EACH ROW

BEGIN

DECLARE var\_stato TINYINT;

SELECT aperto\_or\_finalizzato

FROM ordini

WHERE id = NEW.ordine

INTO var\_stato;

IF var\_stato = 0 THEN

SIGNAL SQLSTATE '45007'

SET MESSAGE\_TEXT = "It is not possible to modify an order already closed";

END IF;

END$$

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`appartenenza\_ordini\_AFTER\_INSERT`

AFTER INSERT ON `appartenenza\_ordini` FOR EACH ROW

BEGIN

UPDATE specie\_di\_piante

SET giacenza = giacenza - NEW.quantita

WHERE codice = NEW.specie\_di\_pianta;

END$$

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`appartenenza\_ordini\_BEFORE\_UPDATE`

BEFORE UPDATE ON `appartenenza\_ordini` FOR EACH ROW

BEGIN

DECLARE var\_giacenza INT UNSIGNED;

SELECT giacenza

FROM specie\_di\_piante

WHERE codice = NEW.specie\_di\_pianta

INTO var\_giacenza;

IF CAST(NEW.quantita AS SIGNED) - CAST(OLD.quantita AS SIGNED) > var\_giacenza

THEN

SIGNAL SQLSTATE '45006'

SET MESSAGE\_TEXT = "Selected quantity is higher than current stock";

END IF;

END$$

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`appartenenza\_ordini\_BEFORE\_UPDATE\_1`

BEFORE UPDATE ON `appartenenza\_ordini` FOR EACH ROW

BEGIN

DECLARE var\_stato TINYINT;

SELECT aperto\_or\_finalizzato

FROM ordini

WHERE id = NEW.ordine

INTO var\_stato;

IF var\_stato = 0 THEN

SIGNAL SQLSTATE '45007'

SET MESSAGE\_TEXT = "It is not possible to modify an order already closed";

END IF;

END$$

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`appartenenza\_ordini\_AFTER\_UPDATE`

AFTER UPDATE ON `appartenenza\_ordini` FOR EACH ROW

BEGIN

IF NEW.quantita > OLD.quantita THEN

UPDATE specie\_di\_piante

SET giacenza = giacenza - (NEW.quantita - OLD.quantita)

WHERE codice = NEW.specie\_di\_pianta;

ELSE

UPDATE specie\_di\_piante

SET giacenza = giacenza + (OLD.quantita - NEW.quantita)

WHERE codice = NEW.specie\_di\_pianta;

END IF;

END$$

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`appartenenza\_ordini\_BEFORE\_DELETE`

BEFORE DELETE ON `appartenenza\_ordini` FOR EACH ROW

BEGIN

DECLARE var\_stato TINYINT;

SELECT aperto\_or\_finalizzato

FROM ordini

WHERE id = OLD.ordine

INTO var\_stato;

IF var\_stato = 0 THEN

SIGNAL SQLSTATE '45007'

SET MESSAGE\_TEXT = "It is not possible to modify an order already closed";

END IF;

END$$

CREATE DEFINER = CURRENT\_USER TRIGGER

`verdesrl`.`appartenenza\_ordini\_AFTER\_DELETE`

AFTER DELETE ON `appartenenza\_ordini` FOR EACH ROW

BEGIN

DECLARE var\_stato TINYINT;

SELECT aperto\_or\_finalizzato

FROM ordini

WHERE id = OLD.ordine

INTO var\_stato;

IF var\_stato = 1 THEN

UPDATE specie\_di\_piante

SET giacenza = giacenza + OLD.quantita

WHERE codice = OLD.specie\_di\_pianta;

END IF;

END$$

## Eventi

Entrambi gli eventi non sono attivati in fase di configurazione bensì vengono eseguiti una volta all’anno per eliminare dati non più utili al sistema.

Per quanto riguarda il secondo evento è stato supposto che lo scarico delle forniture avvenga regolarmente il lunedì sera per cui tale evento viene eseguito ogni martedì a mezzanotte.

Il codice di tali eventi è qui riportato:

SET GLOBAL event\_scheduler = ON;

DELIMITER $$

CREATE EVENT IF NOT EXISTS pulizia\_storico\_dati

ON SCHEDULE EVERY 1 YEAR

STARTS "2021-01-01 00:00:00"

ON COMPLETION PRESERVE

COMMENT "Eliminazione dati relativi ad ordini prezzi e richieste di fornitura memorizzati da piu' di due anni "

DO BEGIN

DELETE FROM `verdesrl`.`ordini`

WHERE `data` < (NOW() - INTERVAL 2 YEAR);

DELETE FROM `verdesrl`.`richieste\_di\_forniture`

WHERE `data` < (NOW() - INTERVAL 2 YEAR);

DELETE FROM `verdesrl`.`prezzi`

WHERE `data\_termine` < (NOW() - INTERVAL 2 YEAR);

END $$

DELIMITER ;

CREATE EVENT IF NOT EXISTS aggiornamento\_giacenza

ON SCHEDULE EVERY 1 WEEK

STARTS "2020-04-21 00:00:00"

ON COMPLETION PRESERVE

COMMENT "Aggiornamento giacenza al fronte di richieste di forniture"

DO BEGIN

DECLARE var\_specie\_c INT;

DECLARE var\_quantita\_c INT;

DECLARE var\_data\_agg DATETIME;

DECLARE done INT DEFAULT FALSE;

DECLARE cur CURSOR FOR

SELECT `specie\_richiesta`, `quantita`

FROM `richieste\_di\_forniture`

WHERE `pendente\_si\_no` = 1;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

DROP TEMPORARY TABLE IF EXISTS da\_aggiornare;

CREATE TEMPORARY TABLE da\_aggiornare (

`specie\_di\_pianta` INT,

`quantita\_richiesta` INT UNSIGNED

);

SET var\_data\_agg = now();

OPEN cur;

read\_loop: LOOP

FETCH cur INTO var\_specie\_c, var\_quantita\_c;

IF done THEN

LEAVE read\_loop;

END IF;

UPDATE `specie\_di\_piante`

SET `giacenza` = `giacenza` + var\_quantita\_c

WHERE `codice` = var\_specie\_c;

END LOOP;

CLOSE cur;

DROP TEMPORARY TABLE da\_aggiornare;

UPDATE `richieste\_di\_forniture`

SET `pendente\_si\_no` = 0

WHERE `data` <= var\_data\_agg;

END $$

DELIMITER ;

## Viste

Non sono state implementate viste

## Stored Procedures e transazioni

**XCS-01**

Il livello di isolamento scelto e' SERIALIZABLE per via dell'utilizzo della select max per ottenere l'id autogereato.

Non e' stato possibile utilizzare la funzione LAST\_INSERT\_ID() poiche' dalla documentazione si evince che "The ID that was generated is maintained in the server on a per-connection basis".

Per cui dato che la connessione e' la medesima per ciascun utente afferente ad una certa classe si e' preferita questa scelta alternativa.

CREATE PROCEDURE `crea\_ordine` (

IN var\_cliente VARCHAR(16),

IN var\_ind\_sped VARCHAR(64),

IN var\_contatto VARCHAR(256),

IN var\_specie INT,

IN var\_quantita INT,

OUT var\_id INT

)

BEGIN

DECLARE var\_errno INT;

DECLARE var\_verifica\_proprietario INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1452 THEN

SIGNAL SQLSTATE '45014'

SET MESSAGE\_TEXT = "Selected species does not exists";

ELSE

RESIGNAL;

END IF;

END;

DROP TABLE IF EXISTS `verifica\_proprietario`;

CREATE TEMPORARY TABLE `verifica\_proprietario` (

`Contatto` VARCHAR(256),

`Cliente` CHAR(16),

`Referente` CHAR(11)

);

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

START TRANSACTION;

IF var\_ind\_sped IS NULL THEN

SELECT indirizzo\_residenza

FROM clienti

WHERE codice = var\_cliente

INTO var\_ind\_sped;

END IF;

IF var\_contatto IS NULL THEN

SELECT contatto\_preferito

FROM clienti

WHERE codice = var\_cliente

INTO var\_contatto;

IF var\_contatto IS NULL THEN

SIGNAL SQLSTATE '45002'

SET MESSAGE\_TEXT = "Favourite contact not set for current user";

END IF;

ELSE

INSERT INTO `verifica\_proprietario`

SELECT contatti.valore AS `Contatto`,

contatti\_clienti.cliente AS `Cliente`,

contatti\_referenti.referente AS `Referente`

FROM contatti LEFT JOIN contatti\_clienti

ON contatti\_clienti.contatto = valore

LEFT JOIN contatti\_referenti

ON contatti\_referenti.contatto = valore

WHERE contatti.valore = var\_contatto;

SELECT COUNT(\*)

FROM `verifica\_proprietario`

WHERE (`Cliente` = var\_cliente OR `Referente` = var\_cliente)

INTO var\_verifica\_proprietario;

IF var\_verifica\_proprietario = 0 THEN

SIGNAL SQLSTATE '45009'

SET MESSAGE\_TEXT = "Neither customer nor referent

(if exists) own this contact";

END IF;

END IF;

DROP TEMPORARY TABLE `verifica\_proprietario`;

INSERT INTO ordini (`cliente`, `data`, `aperto\_or\_finalizzato`,

`indirizzo\_spedizione`, `contatto`)

VALUES (var\_cliente, NOW(), 1, var\_ind\_sped, var\_contatto);

SELECT MAX(id) FROM ordini INTO var\_id;

INSERT INTO appartenenza\_ordini (`specie\_di\_pianta`, `ordine`, `quantita`)

VALUES (var\_specie, var\_id, var\_quantita);

COMMIT;

END$$

**XCS-02**

Il livello di isolamento scelto e' REPEATABLE READ, al fine di garantire che la giacenza della specie (che viene letta in un trigger BI) non possa essere modificata durante l'esecuzione della transazione.

CREATE PROCEDURE `aggiungi\_specie\_ad\_ordine\_esistente` (

IN var\_cliente VARCHAR(16),

IN var\_specie INT,

IN var\_ordine INT,

IN var\_quantita INT

)

BEGIN

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1062 THEN

SIGNAL SQLSTATE '45008'

SET MESSAGE\_TEXT = "Species already in order";

ELSEIF var\_errno = 1452 THEN

SIGNAL SQLSTATE '45014'

SET MESSAGE\_TEXT = "Species does not exists";

ELSE

RESIGNAL;

END IF;

END;

START TRANSACTION;

IF verifica\_proprietario(var\_ordine, var\_cliente) THEN

INSERT INTO appartenenza\_ordini (`specie\_di\_pianta`, `ordine`, `quantita`)

VALUES (var\_specie, var\_ordine, var\_quantita);

ELSE

SIGNAL SQLSTATE '45005'

SET MESSAGE\_TEXT = "Only owners can make changes to an order";

END IF;

COMMIT;

END$$

**XCS-03**

Il livello di isolamento scelto è SERIALIZABLE per via dell’utilizzo della funzione COUNT(\*) per contare le specie ancora appartenenti all’ordine. Ai fini del corretto funzionamento è necessario che non vi siano inserimenti fantasma fino al termine della transazione.

CREATE PROCEDURE `rimuovi\_specie\_da\_ordine` (

IN var\_cliente VARCHAR(16),

IN var\_specie INT,

IN var\_ordine INT,

OUT var\_ordine\_eliminato\_si\_no INT,

OUT var\_eliminazione\_eff INT

)

BEGIN

DECLARE var\_capienza\_ordine INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

RESIGNAL;

END;

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

START TRANSACTION;

IF verifica\_proprietario(var\_ordine, var\_cliente) THEN

SELECT COUNT(\*)

FROM appartenenza\_ordini

WHERE specie\_di\_pianta = var\_specie AND

ordine = var\_ordine

INTO var\_eliminazione\_eff;

IF var\_eliminazione\_eff = 1 THEN

DELETE

FROM appartenenza\_ordini

WHERE specie\_di\_pianta = var\_specie AND

ordine = var\_ordine;

SELECT COUNT(\*)

FROM appartenenza\_ordini

WHERE ordine = var\_ordine

INTO var\_capienza\_ordine;

IF var\_capienza\_ordine = 0 THEN

DELETE FROM ordini WHERE id = var\_ordine;

SET var\_ordine\_eliminato\_si\_no = 1;

ELSE

SET var\_ordine\_eliminato\_si\_no = 0;

END IF;

END IF;

ELSE

SIGNAL SQLSTATE '45005'

SET MESSAGE\_TEXT = "Only owners can make changes to an order";

END IF;

COMMIT;

END$$

**XCS-04**

Il livello di isolamento scelto e' REPEATABLE READ, al fine di garantire che la giacenza della specie (che viene letta in un trigger BU) non possa essere modificata durante l'esecuzione della transazione.

CREATE PROCEDURE `modifica\_ordine` (

IN var\_cliente VARCHAR(16),

IN var\_specie INT,

IN var\_ordine INT,

IN var\_quantita INT,

OUT var\_aggiornamento\_eff INT

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

RESIGNAL;

END;

START TRANSACTION;

IF verifica\_proprietario(var\_ordine, var\_cliente) THEN

SELECT COUNT(\*)

FROM appartenenza\_ordini

WHERE specie\_di\_pianta = var\_specie AND

ordine = var\_ordine

INTO var\_aggiornamento\_eff;

IF var\_aggiornamento\_eff = 1 THEN

UPDATE appartenenza\_ordini

SET quantita = var\_quantita

WHERE specie\_di\_pianta = var\_specie AND

ordine = var\_ordine;

END IF;

ELSE

SIGNAL SQLSTATE '45005'

SET MESSAGE\_TEXT = "Only owners can make changes to an order";

END IF;

COMMIT;

END$$

**XCS-05**

CREATE PROCEDURE `finalizza\_ordine` (

IN var\_cliente VARCHAR(16),

IN var\_ordine INT

)

BEGIN

IF verifica\_proprietario(var\_ordine, var\_cliente) THEN

UPDATE ordini

SET aperto\_or\_finalizzato = 0

WHERE id = var\_ordine;

ELSE

SIGNAL SQLSTATE '45005'

SET MESSAGE\_TEXT = "Only owners can make changes to an order";

END IF;

END$$

**XCS-06**

Il livello di isolamento scelto e' SERIALIZABLE alfine di presentare uno snapshot affidabile dell'ordineselezionato anche nel caso in cui esso sia ancora apertoe quindi potrebbe mutare concorrentemente alla produzionedel report con eventuali inserimenti.

Il campo nome all’interno della tabella temporanea dettagli\_specie tiene conto della somma di più lunghezze in particolare si ha 131 = 64 (nome\_comune) + 64 (nome\_latino) + 3 (caratteri extra).

CREATE PROCEDURE `report\_ordine` (

IN var\_cliente VARCHAR(16),

IN var\_ordine INT

)

BEGIN

DECLARE var\_codice\_cursor INT;

DECLARE var\_verifica\_esistenza INT DEFAULT 0;

DECLARE var\_data\_ordine DATETIME;

DECLARE var\_numero\_piante INT;

DECLARE done INT DEFAULT FALSE;

DECLARE cur CURSOR FOR

SELECT codice

FROM specie\_di\_piante INNER JOIN appartenenza\_ordini

ON codice = specie\_di\_pianta

WHERE ordine = var\_ordine;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

RESIGNAL;

END;

DROP TEMPORARY TABLE IF EXISTS dettagli\_specie;

CREATE TEMPORARY TABLE dettagli\_specie (

`Code` INT,

`Name` VARCHAR(131),

`Unit\_price\_(E)` DECIMAL(7, 2),

`Required\_amount` INT,

`Cumulative\_price\_(E)` DECIMAL(8, 2),

PRIMARY KEY (`Code`)

);

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

START TRANSACTION;

IF NOT verifica\_proprietario(var\_ordine, var\_cliente) THEN

SIGNAL SQLSTATE '45005'

SET MESSAGE\_TEXT = "Only owners can view order reports";

END IF;

SELECT COUNT(\*)

FROM ordini

WHERE id = var\_ordine

INTO var\_verifica\_esistenza;

IF var\_verifica\_esistenza = 0 THEN

SIGNAL SQLSTATE '45016'

SET MESSAGE\_TEXT = "Not existent order";

END IF;

SELECT `data`

FROM `ordini`

WHERE `id` = var\_ordine

INTO var\_data\_ordine;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO var\_codice\_cursor;

IF done THEN

LEAVE read\_loop;

END IF;

INSERT INTO dettagli\_specie

SELECT specie\_di\_piante.codice AS `Code`,

CONCAT(specie\_di\_piante.nome\_comune, " (",

specie\_di\_piante.nome\_latino, + ")") AS `Name`,

prezzi.valore AS `Unit\_price\_(E)`,

appartenenza\_ordini.quantita AS `Required\_amount`,

prezzi.valore \* appartenenza\_ordini.quantita AS `Cumulative\_price\_(E)`

FROM specie\_di\_piante INNER JOIN prezzi

ON specie\_di\_piante.codice = prezzi.specie\_di\_pianta

INNER JOIN appartenenza\_ordini ON

specie\_di\_piante.codice=appartenenza\_ordini.specie\_di\_pianta

WHERE specie\_di\_piante.codice = var\_codice\_cursor AND

data\_adozione <= var\_data\_ordine

ORDER BY data\_adozione DESC

LIMIT 1;

END LOOP;

CLOSE cur;

SELECT `ordini`.`id` AS `ID`,

DATE(`ordini`.`data`) AS `Date`,

`ordini`.`indirizzo\_spedizione` AS `Shipping\_address`,

`ordini`.`contatto` AS `Chosen\_contact`,

CASE

WHEN `ordini`.`aperto\_or\_finalizzato` = 1 THEN "Open"

ELSE "Finalized"

END AS `Status`,

CASE

WHEN `clienti`.`privato\_or\_rivendita` = 1 THEN

"Not expected"

ELSE concat(`referenti`.`nome`, " ", `referenti`.`cognome`)

END AS `Referent`

FROM `ordini` INNER JOIN clienti` ON `ordini`.`cliente` = `clienti`.`codice`

LEFT JOIN `referenti` ON `clienti`.`codice` = `referenti`.`rivendita`

WHERE `id` = var\_ordine;

SELECT \* FROM dettagli\_specie;

SELECT SUM(`Required\_amount`) AS `Number\_of\_plants`,

SUM(`Cumulative\_price\_(E)`) AS `Total\_spending`

FROM `dettagli\_specie`;

DROP TEMPORARY TABLE dettagli\_specie;

COMMIT;

END$$

**XCS-07**

CREATE PROCEDURE `modifica\_residenza` (

IN var\_cliente VARCHAR(16),

IN var\_residenza VARCHAR(64)

)

BEGIN

UPDATE clienti

SET indirizzo\_residenza = var\_residenza

WHERE codice = var\_cliente;

END$$

**XCS-08**

CREATE PROCEDURE `modifica\_fatturazione` (

IN var\_cliente VARCHAR(16),

IN var\_fatturazione VARCHAR(64)

)

BEGIN

UPDATE clienti

SET indirizzo\_fatturazione = var\_fatturazione

WHERE codice = var\_cliente;

END$$

**XCS-09**

Il livello di isolamento scelto e' READ COMMITTED poichè sufficiente per restituire una lista consistente di contatti realmente inseriti (un contatto già inserito non può essere modificato).

CREATE PROCEDURE `aggiungi\_contatto\_cliente` (

IN var\_cliente VARCHAR(16),

IN var\_contatto VARCHAR(256),

IN var\_tipo CHAR(16)

)

BEGIN

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1062 THEN

SIGNAL SQLSTATE '45010'

SET MESSAGE\_TEXT = "Contact already inserted";

ELSE

RESIGNAL;

END IF;

END;

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

START TRANSACTION;

INSERT INTO contatti (`valore`, `tipo`) VALUES (var\_contatto, var\_tipo);

INSERT INTO contatti\_clienti (`contatto`, `cliente`)

VALUES (var\_contatto, var\_cliente);

SELECT contatti.valore AS `Contact`,

contatti.tipo AS `Type`

FROM contatti INNER JOIN contatti\_clienti

ON contatti.valore = contatti\_clienti.contatto

WHERE cliente = var\_cliente;

COMMIT;

END$$

**XCS-10**

CREATE PROCEDURE `rimuovi\_contatto\_cliente` (

IN var\_cliente VARCHAR(16),

IN var\_contatto VARCHAR(256)

)

BEGIN

DECLARE var\_verifica\_possesso INT;

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1451 THEN

SIGNAL SQLSTATE '45012'

SET MESSAGE\_TEXT = "Cannot remove the contact because it is associated

to an order";

ELSE

RESIGNAL;

END IF;

END;

SELECT COUNT(\*)

FROM contatti\_clienti

WHERE cliente = var\_cliente AND

contatto = var\_contatto

INTO var\_verifica\_possesso;

IF var\_verifica\_possesso = 0 THEN

SIGNAL SQLSTATE '45009'

SET MESSAGE\_TEXT = "Selected customer does not own this contact";

ELSE

DELETE FROM contatti WHERE valore = var\_contatto;

END IF;

END$$

**XCS-11**

CREATE PROCEDURE `modifica\_contatto\_preferito\_cliente` (

IN var\_cliente VARCHAR(16),

IN var\_contatto VARCHAR(256)

)

BEGIN

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1452 THEN

SIGNAL SQLSTATE '45013'

SET MESSAGE\_TEXT = "Cannot set this contact as favourite one because it

is not owned yet";

ELSE

RESIGNAL;

END IF;

END;

UPDATE clienti

SET contatto\_preferito = var\_contatto

WHERE codice = var\_cliente;

END$$

**XCS-12**

CREATE PROCEDURE `visualizza\_contatti\_cliente` (IN var\_cliente VARCHAR(16))

BEGIN

SELECT contatti.valore AS `Contact`,

contatti.tipo AS `Type`

FROM contatti INNER JOIN contatti\_clienti

ON contatti.valore = contatti\_clienti.contatto

WHERE cliente = var\_cliente;

END$$

**XCS-13**

Il livello di isolamento scelto e' READ COMMITTED perché l’unico obiettivo è quello di leggere l’informazione relativa allo stato (aperto/finalizzato) di un ordine in maniera consistente. Dato che essa non viene letta nuovamente o utilizzata in altre operazioni non è necessario un livello di isolamento più alto.

CREATE PROCEDURE `visualizza\_ordini\_cliente` (

IN var\_cliente VARCHAR(16),

IN var\_status TINYINT

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

RESIGNAL;

END;

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

START TRANSACTION;

IF var\_status = 1 THEN

SELECT `id` AS `Order`,

`indirizzo\_spedizione` AS `Shipping\_address`,

DATE(`data`) AS `Date`

FROM `ordini`

WHERE `cliente` = var\_cliente AND

`aperto\_or\_finalizzato` = 1

ORDER BY `data` DESC;

ELSE

SELECT `id` AS `Order`,

`indirizzo\_spedizione` AS `Shipping address`,

DATE(`data`) AS `Date`,

CASE

WHEN `aperto\_or\_finalizzato` = 1 THEN "Open"

ELSE "Finalized"

END AS `Status`

FROM `ordini`

WHERE `cliente` = var\_cliente

ORDER BY `data` DESC;

END IF;

COMMIT;

END$$

**XCS-14**

Il livello di isolamento scelto e' READ COMMITTED perché l’unico obiettivo è quello di leggere l’informazione relativa alla giacenza relativa ad una certa specie in maniera consistente. Dato che essa non viene letta nuovamente o utilizzata in altre operazioni non è necessario un livello di isolamento più alto.

CREATE PROCEDURE `visualizza\_specie\_appartenenti\_ad\_ordine` (IN var\_ordine INT)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

RESIGNAL;

END;

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

START TRANSACTION;

SELECT `codice` AS `Species\_code`,

CONCAT(`nome\_comune`, " (", `nome\_latino`, + ")")

AS `Species\_name`,

`quantita` AS `Quantity`

FROM `appartenenza\_ordini` INNER JOIN `specie\_di\_piante` ON

`specie\_di\_pianta` = `codice`

WHERE `ordine` = var\_ordine;

COMMIT;

END$$

**XWM-01**

Il livello di isolamento scelto e' READ COMMITTED perché l’unico obiettivo è quello di leggere l’informazione relativa alla giacenza relativa ad una certa specie in maniera consistente. Dato che essa non viene letta nuovamente o utilizzata in altre operazioni non è necessario un livello di isolamento più alto.

CREATE PROCEDURE `visualizza\_dettagli\_specie` (

IN var\_nome\_comune VARCHAR(64),

IN var\_solo\_fiorite TINYINT

)

BEGIN

DECLARE var\_ricerca VARCHAR(66);

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

RESIGNAL;

END;

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

START TRANSACTION;

IF var\_nome\_comune IS NOT NULL THEN

SET var\_ricerca = CONCAT("%", UPPER(var\_nome\_comune), "%");

ELSE

SET var\_ricerca = "%";

END IF;

IF var\_solo\_fiorite = 0 THEN

SELECT codice AS `Code`,

nome\_latino AS `Latin\_name`,

nome\_comune AS `Common\_name`,

CASE

WHEN interni\_or\_esterni = 1 THEN 'Indoor'

ELSE 'Outdoor'

END AS `Indoor/Outdoor`,

CASE

WHEN verde\_or\_fiorita = 1 THEN 'Green'

ELSE 'Flowery'

END AS `Green/Flowery`,

CASE

WHEN esotica\_si\_no = 1 THEN 'Yes'

ELSE 'No'

END AS `Exotic`,

valore AS `Price`,

giacenza AS `Stock`

FROM specie\_di\_piante INNER JOIN prezzi

ON specie\_di\_piante.codice = prezzi.specie\_di\_pianta

WHERE UPPER(nome\_comune) LIKE var\_ricerca AND

data\_termine IS NULL

ORDER BY nome\_comune ASC;

ELSE

SELECT codice AS `Code`,

nome\_latino AS `Latin name`,

nome\_comune AS `Common name`,

CASE

WHEN interni\_or\_esterni = 1 THEN 'Indoor'

ELSE 'Outdoor'

END AS `Indoor/Outdoor`,

CASE

WHEN verde\_or\_fiorita = 1 THEN 'Green'

ELSE 'Flowery'

END AS `Green/Flowery`,

CASE

WHEN esotica\_si\_no = 1 THEN 'Yes'

ELSE 'No'

END AS `Exotic`,

valore AS `Price`,

giacenza AS `Stock`

FROM specie\_di\_piante INNER JOIN prezzi

ON specie\_di\_piante.codice = prezzi.specie\_di\_pianta

WHERE UPPER(nome\_comune) LIKE var\_ricerca AND

data\_termine IS NULL AND

verde\_or\_fiorita = 0

ORDER BY nome\_comune ASC;

END IF;

COMMIT;

END$$

**RCS-01**

Il livello di isolamento scelto e' READ COMMITTED poichè sufficiente per restituire una lista consistente di contatti realmente inseriti (un contatto già inserito non può essere modificato).

CREATE PROCEDURE `aggiungi\_contatto\_referente` (

IN var\_referente CHAR(11),

IN var\_contatto VARCHAR(256),

IN var\_tipo CHAR(16)

)

BEGIN

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1062 THEN

SIGNAL SQLSTATE '45010'

SET MESSAGE\_TEXT = "Contact already inserted";

ELSE

RESIGNAL;

END IF;

END;

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

START TRANSACTION;

INSERT INTO contatti (`valore`, `tipo`) VALUES (var\_contatto, var\_tipo);

INSERT INTO contatti\_referenti (`contatto`, `referente`)

VALUES (var\_contatto, var\_referente);

SELECT contatti.valore AS `Contact`,

contatti.tipo AS `Type`

FROM contatti INNER JOIN contatti\_referenti

ON contatti.valore = contatti\_referenti.contatto

WHERE referente = var\_referente;

COMMIT;

END$$

**RCS-02**

CREATE PROCEDURE `rimuovi\_contatto\_referente` (

IN var\_referente CHAR(11),

IN var\_contatto VARCHAR(256)

)

BEGIN

DECLARE var\_verifica\_possesso INT;

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1451 THEN

SIGNAL SQLSTATE '45012'

SET MESSAGE\_TEXT = "Cannot remove the contact because it is associated

to an order";

ELSE

RESIGNAL;

END IF;

END;

SELECT COUNT(\*)

FROM contatti\_referenti

WHERE referente = var\_referente AND

contatto = var\_contatto

INTO var\_verifica\_possesso;

IF var\_verifica\_possesso = 0 THEN

SIGNAL SQLSTATE '45009'

SET MESSAGE\_TEXT = "Selected referent does not own this contact";

ELSE

DELETE FROM contatti WHERE valore = var\_contatto;

END IF;

END$$

**RCS-03**

CREATE PROCEDURE `modifica\_contatto\_preferito\_referente` (

IN var\_referente CHAR(11),

IN var\_contatto VARCHAR(256)

)

BEGIN

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1452 THEN

SIGNAL SQLSTATE '45013'

SET MESSAGE\_TEXT = "Cannot set this contact as favourite one because it

is not owned yet";

ELSE

RESIGNAL;

END IF;

END;

UPDATE referenti

SET contatto\_preferito = var\_contatto

WHERE rivendita = var\_referente;

END$$

**RCS-04**

CREATE PROCEDURE `visualizza\_contatti\_referente` (IN var\_referente VARCHAR(16))

BEGIN

SELECT contatti.valore AS `Contact`,

contatti.tipo AS `Type`

FROM contatti INNER JOIN contatti\_referenti

ON contatti.valore = contatti\_referenti.contatto

WHERE referente = var\_referente;

END$$

**MNG-01**

Il livello di isolamento scelto e' SERIALIZABLE per via dell'utilizzo della select max per ottenere l'id autogereato.

Non e' stato possibile utilizzare la funzione LAST\_INSERT\_ID() poiche' dalla documentazione si evince che "The ID that was generated is maintained in the server on a per-connection basis".

Per cui dato che la connessione e' la medesima per ciascun utente afferente ad una certa classe si e' preferita questa scelta alternativa.

CREATE PROCEDURE `inserisci\_nuova\_specie` (

IN var\_nome\_comune VARCHAR(64),

IN var\_nome\_latino VARCHAR(64),

IN var\_int\_est TINYINT,

IN var\_colorazione VARCHAR(32),

IN var\_esotica TINYINT,

IN var\_giacenza INT,

IN var\_prezzo DECIMAL(7, 2),

OUT var\_codice INT

)

BEGIN

DECLARE var\_verde\_or\_fiorita TINYINT;

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1062 THEN

SIGNAL SQLSTATE '45021'

SET MESSAGE\_TEXT = "Species already stored";

ELSE

RESIGNAL;

END IF;

END;

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

START TRANSACTION;

IF var\_colorazione IS NULL THEN

SET var\_verde\_or\_fiorita = 1;

ELSE

SET var\_verde\_or\_fiorita = 0;

END IF;

INSERT INTO specie\_di\_piante (nome\_comune, nome\_latino, interni\_or\_esterni,

verde\_or\_fiorita, esotica\_si\_no, giacenza)

VALUES (var\_nome\_comune, var\_nome\_latino, var\_int\_est,

var\_verde\_or\_fiorita, var\_esotica, var\_giacenza);

SELECT MAX(codice) FROM specie\_di\_piante INTO var\_codice;

IF var\_colorazione IS NOT NULL THEN

INSERT INTO colorazioni (specie\_fiorita, colore)

VALUES (var\_codice, var\_colorazione);

END IF;

INSERT INTO prezzi (specie\_di\_pianta, data\_adozione, valore)

VALUES (var\_codice, NOW(), var\_prezzo);

COMMIT;

END$$

**MNG-02**

CREATE PROCEDURE `rimuovi\_specie\_di\_pianta` (

IN var\_codice INT,

OUT var\_eliminazione\_effettiva INT

)

BEGIN

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1451 THEN

SIGNAL SQLSTATE '45029'

SET MESSAGE\_TEXT = "Cannot remove the species because it is associated

to an order";

ELSE

RESIGNAL;

END IF;

END;

SELECT COUNT(\*)

FROM specie\_di\_piante

WHERE codice = var\_codice

INTO var\_eliminazione\_effettiva;

IF var\_eliminazione\_effettiva = 1 THEN

DELETE FROM specie\_di\_piante WHERE codice = var\_codice;

END IF;

END$$

**MNG-03**

In questo caso l’operazione non utilizza una transazione se pure il suo schema è analogo alle stored procedure di inserimento contatti. In quei casi però la transazione è indispensabile perché vengono effettuate due scritture le quali devono essere eseguite atomicamente.

In questo caso la scrittura è una sola e non vale la pena introdurre del overhead dovuto all’introduzione di una transazione soltanto per l’output della lista aggiornata (al più infatti essa può contenere un qualche colore in più che in realtà non dovrebbe esser presente).

CREATE PROCEDURE `aggiungi\_colorazione` (

IN var\_specie\_fiorita INT,

IN var\_colore VARCHAR(32)

)

BEGIN

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1452 THEN

SIGNAL SQLSTATE '45014'

SET MESSAGE\_TEXT = "Species does not exists";

ELSEIF var\_errno = 1062 THEN

SIGNAL SQLSTATE '45030'

SET MESSAGE\_TEXT = "Coloring already specified for selected species";

ELSE

RESIGNAL;

END IF;

END;

INSERT INTO colorazioni (`specie\_fiorita`, `colore`)

VALUES (var\_specie\_fiorita, var\_colore);

SELECT UPPER(colore) AS `Coloring`

FROM colorazioni

WHERE specie\_fiorita = var\_specie\_fiorita;

END$$

**MNG-04**

CREATE PROCEDURE `rimuovi\_colorazione` (

IN var\_specie\_fiorita INT,

IN var\_colore VARCHAR(32),

OUT var\_eliminazione\_effettiva INT

)

BEGIN

SELECT COUNT(\*)

FROM colorazioni

WHERE specie\_fiorita = var\_specie\_fiorita AND

colore = var\_colore

INTO var\_eliminazione\_effettiva;

IF var\_eliminazione\_effettiva = 1 THEN

DELETE

FROM colorazioni

WHERE specie\_fiorita = var\_specie\_fiorita AND

colore = var\_colore;

END IF;

END$$

**MNG-05**

Il livello scelto in questa transazione e' READ COMMITTED poiche' l'unica necessita' e' quella di leggere l'ultimo prezzo adottato in maniera consistente.

Viene inoltre fattorizzata l'impostazione della data per non aver nessuna differenza fra quella di terminazione e quella di nuova adozione

CREATE PROCEDURE `modifica\_prezzo` (

IN var\_specie INT,

IN var\_prezzo DECIMAL(7, 2),

OUT var\_aggiornamento\_effettivo INT

)

BEGIN

DECLARE var\_nuova\_data DATETIME;

DECLARE var\_prezzo\_attuale DECIMAL(7, 2);

DECLARE var\_specie\_non\_trovata TINYINT DEFAULT 0;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

RESIGNAL;

END;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET var\_specie\_non\_trovata = 1;

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

START TRANSACTION;

SELECT valore

FROM prezzi

WHERE specie\_di\_pianta = var\_specie AND

data\_termine IS NULL

INTO var\_prezzo\_attuale;

IF var\_specie\_non\_trovata = 1 THEN

SIGNAL SQLSTATE '45014'

SET MESSAGE\_TEXT = "Species does not exists";

END IF;

IF var\_prezzo\_attuale <> var\_prezzo THEN

SET var\_nuova\_data = NOW();

UPDATE prezzi

SET data\_termine = var\_nuova\_data

WHERE specie\_di\_pianta = var\_specie AND

data\_termine IS NULL;

INSERT INTO prezzi (specie\_di\_pianta, data\_adozione, valore)

VALUES (var\_specie, var\_nuova\_data, var\_prezzo);

SET var\_aggiornamento\_effettivo = 1;

ELSE

SET var\_aggiornamento\_effettivo = 0;

END IF;

COMMIT;

END$$

**MNG-06**

Il livello di isolamento scelto e' REPEATABLE READ poichè si suppone che le vendite avvenute in concorrenza con l’esecuzione di tale stored procedure (che quindi andrebbero in taluni casi ad aggiungere nuovi record alle tabella appartenenza\_ordini) non possono influenzare fortemente un report su base mensile.

CREATE PROCEDURE `report\_specie` (IN var\_codice INT)

BEGIN

DECLARE var\_nome VARCHAR(131);

DECLARE var\_mese INT DEFAULT 1;

DECLARE var\_mese\_string VARCHAR(16);

DECLARE var\_vendite INT;

DECLARE var\_verifica\_esistenza INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

RESIGNAL;

END;

DROP TEMPORARY TABLE IF EXISTS `report\_mensile`;

CREATE TEMPORARY TABLE `report\_mensile` (

`Month` VARCHAR(16),

`Sales` INT

);

START TRANSACTION;

SELECT COUNT(\*)

FROM specie\_di\_piante

WHERE codice = var\_codice

INTO var\_verifica\_esistenza;

IF var\_verifica\_esistenza <> 1 THEN

SIGNAL SQLSTATE '45014'

SET MESSAGE\_TEXT = "Species does not exists";

END IF;

SELECT codice AS `Species\_code`,

nome\_comune AS `Common\_name`,

nome\_latino AS `Latin\_name`

FROM specie\_di\_piante

WHERE codice = var\_codice;

for\_loop: LOOP

IF var\_mese = 13 THEN

LEAVE for\_loop;

END IF;

SELECT SUM(`appartenenza\_ordini`.`quantita`)

FROM `appartenenza\_ordini` INNER JOIN `ordini`

ON `appartenenza\_ordini`.`ordine` = `ordini`.`id`

WHERE `aperto\_or\_finalizzato` = 0 AND

YEAR(`data`) = YEAR(CURDATE()) AND

MONTH(`data`) = var\_mese AND

`specie\_di\_pianta` = var\_codice

INTO var\_vendite;

IF var\_vendite IS NULL THEN

SET var\_vendite = 0;

END IF;

CASE var\_mese

WHEN 1 THEN SET var\_mese\_string = "January";

WHEN 2 THEN SET var\_mese\_string = "February";

WHEN 3 THEN SET var\_mese\_string = "March";

WHEN 4 THEN SET var\_mese\_string = "April";

WHEN 5 THEN SET var\_mese\_string = "May";

WHEN 6 THEN SET var\_mese\_string = "June";

WHEN 7 THEN SET var\_mese\_string = "July";

WHEN 8 THEN SET var\_mese\_string = "August";

WHEN 9 THEN SET var\_mese\_string = "September";

WHEN 10 THEN SET var\_mese\_string = "October";

WHEN 11 THEN SET var\_mese\_string = "November";

WHEN 12 THEN SET var\_mese\_string = "December";

ELSE SET var\_mese\_string = "####";

END CASE;

INSERT INTO `report\_mensile` (`Month`, `Sales`)

VALUES (var\_mese\_string, var\_vendite);

SET var\_mese = var\_mese + 1;

END LOOP;

SELECT \* FROM `report\_mensile`;

DROP TEMPORARY TABLE `report\_mensile`;

COMMIT;

END$$

**MNG-07**

CREATE PROCEDURE `visualizza\_colorazioni` (IN var\_specie\_fiorita INT)

BEGIN

SELECT UPPER(colore) AS `Coloring`

FROM colorazioni

WHERE specie\_fiorita = var\_specie\_fiorita;

END$$

**WHC-01**

CREATE PROCEDURE `inserisci\_richiesta\_fornitura` (

IN var\_fornitore INT,

IN var\_specie INT,

IN var\_quantita INT

)

BEGIN

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1452 THEN

SIGNAL SQLSTATE '45025'

SET MESSAGE\_TEXT = "Either species or supplier does not exists";

ELSE

RESIGNAL;

END IF;

END;

INSERT INTO richieste\_di\_forniture (`data`, `fornitore\_scelto`, `specie\_richiesta`,

`quantita`)

VALUES (NOW(), var\_fornitore, var\_specie, var\_quantita);

SELECT DATE(`richieste\_di\_forniture`.`data`) AS `Date`,

`fornitori`.`codice\_fornitore` AS `Supplier\_code`,

`fornitori`.`nome` AS `Supplier\_name`,

`richieste\_di\_forniture`.`quantita` AS `Quantity`

FROM `richieste\_di\_forniture` INNER JOIN `fornitori`

ON `richieste\_di\_forniture`.`fornitore\_scelto` = `fornitori`.`codice\_fornitore`

WHERE `richieste\_di\_forniture`.`specie\_richiesta` = var\_specie AND

`richieste\_di\_forniture`.`pendente\_si\_no` = 1

ORDER BY `richieste\_di\_forniture`.`data` DESC;

END$$

**WHC-02**

Il livello di isolamento scelto e' SERIALIZABLE per via dell'utilizzo della select max per ottenere l'id autogereato.

Non e' stato possibile utilizzare la funzione LAST\_INSERT\_ID() poiche' dalla documentazione si evince che "The ID that was generated is maintained in the server on a per-connection basis".

Per cui dato che la connessione e' la medesima per ciascun utente afferente ad una certa classe si e' preferita questa scelta alternativa.

CREATE PROCEDURE `inserisci\_fornitore` (

IN var\_codice\_fiscale CHAR(16),

IN var\_nome VARCHAR(32),

IN var\_specie INT,

IN var\_indirizzo VARCHAR(64),

OUT var\_codice\_fornitore INT

)

BEGIN

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1062 THEN

SIGNAL SQLSTATE '45035'

SET MESSAGE\_TEXT = "Supplier already inserted";

ELSE

RESIGNAL;

END IF;

END;

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

START TRANSACTION;

INSERT INTO fornitori (`codice\_fiscale`, `nome`)

VALUES (var\_codice\_fiscale, var\_nome);

SELECT MAX(codice\_fornitore) FROM fornitori INTO var\_codice\_fornitore;

INSERT INTO disponibilita\_forniture (`fornitore`, `specie\_di\_pianta`)

VALUES (var\_codice\_fornitore, var\_specie);

INSERT INTO indirizzi\_fornitori (`indirizzo`, `fornitore`)

VALUES (var\_indirizzo, var\_codice\_fornitore);

COMMIT;

END$$

**WHC-03**

Anche in questo caso vale lo stesso discorso fatto per l’operazione MNG-03 per quanto riguarda l’assenza di una transazione

CREATE PROCEDURE `aggiungi\_disponibilita\_fornitura` (

IN var\_fornitore INT,

IN var\_specie INT

)

BEGIN

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1452 THEN

SIGNAL SQLSTATE '45025'

SET MESSAGE\_TEXT = "Either species or supplier does not exists";

ELSEIF var\_errno = 1062 THEN

SIGNAL SQLSTATE '45018'

SET MESSAGE\_TEXT = "Supply availability already inserted";

ELSE

RESIGNAL;

END IF;

END;

INSERT INTO disponibilita\_forniture (`fornitore`, `specie\_di\_pianta`)

VALUES (var\_fornitore, var\_specie);

SELECT codice AS `Species\_code`,

nome\_comune AS `Species\_common\_name`,

nome\_latino AS `Species\_latin\_name`

FROM disponibilita\_forniture INNER JOIN specie\_di\_piante

ON codice = specie\_di\_pianta

WHERE fornitore = var\_fornitore;

END$$

**WHC-04**

Il livello di isolamento scelto è REPEATABLE READ poiché sufficiente affinchè i dati presentati nel report siano affidabili. Non è necessario infatti evitare eventuali inserimenti fantasma in questo caso. Si tratterebbe infatti dell’aggiunta di una nuova specie e si può supporre che appena inserita nel sistema non richieda fornitura immediata.

CREATE PROCEDURE `report\_giacenza` (IN var\_range INT)

BEGIN

DECLARE var\_codice\_cursor INT;

DECLARE done INT DEFAULT FALSE;

DECLARE cur CURSOR FOR

SELECT codice

FROM specie\_di\_piante

ORDER BY giacenza ASC

LIMIT var\_range;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

RESIGNAL;

END;

START TRANSACTION;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO var\_codice\_cursor;

IF done THEN

LEAVE read\_loop;

END IF;

SELECT codice AS `Species\_code`,

CONCAT(nome\_comune, " (", nome\_latino, ")") AS `Species\_name`,

giacenza AS `Stock`

FROM specie\_di\_piante

WHERE codice = var\_codice\_cursor;

SELECT codice\_fornitore AS `Supplier\_code`,

nome AS `Supplier\_name`,

COUNT(fornitore\_scelto) AS `No\_request\_made`

FROM fornitori INNER JOIN disponibilita\_forniture

ON codice\_fornitore = fornitore

LEFT JOIN richieste\_di\_forniture

ON fornitore\_scelto = codice\_fornitore

WHERE specie\_di\_pianta = var\_codice\_cursor AND

(specie\_richiesta = var\_codice\_cursor OR specie\_richiesta IS NULL)

GROUP BY `Supplier\_code`, `Supplier\_name`

ORDER BY `No\_request\_made` DESC

LIMIT 5;

END LOOP;

CLOSE cur;

COMMIT;

END$$

**WHC-05**

CREATE PROCEDURE `visualizza\_specie\_disponibili` (IN var\_fornitore INT)

BEGIN

SELECT codice AS `Code`,

nome\_latino AS `Latin\_name`,

nome\_comune AS `Common\_name`

FROM specie\_di\_piante INNER JOIN disponibilita\_forniture

ON codice = specie\_di\_pianta

WHERE fornitore = var\_fornitore

ORDER BY nome\_comune ASC;

END$$

**WHC-06**

Anche in questo caso vale lo stesso discorso fatto per l’operazione MNG-03 per quanto riguarda l’assenza di una transazione

CREATE PROCEDURE `aggiungi\_indirizzo\_fornitore` (

IN var\_fornitore INT,

IN var\_indirizzo VARCHAR(64)

)

BEGIN

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1452 THEN

SIGNAL SQLSTATE '45032'

SET MESSAGE\_TEXT = "Supplier does not exists";

ELSEIF var\_errno = 1062 THEN

SIGNAL SQLSTATE '45031'

SET MESSAGE\_TEXT = "Address already inserted";

ELSE

RESIGNAL;

END IF;

END;

INSERT INTO indirizzi\_fornitori (`indirizzo`, `fornitore`)

VALUES (var\_indirizzo, var\_fornitore);

SELECT UPPER(indirizzo) AS `Address`

FROM indirizzi\_fornitori

WHERE fornitore = var\_fornitore;

END$$

**WHC-07**

CREATE PROCEDURE `visualizza\_fornitori` (IN var\_nome VARCHAR(32))

BEGIN

DECLARE var\_ricerca VARCHAR(34);

IF var\_nome IS NOT NULL THEN

SET var\_ricerca = CONCAT("%", var\_nome, "%");

ELSE

SET var\_ricerca = "%";

END IF;

SELECT codice\_fornitore AS `Supplier\_code`,

codice\_fiscale AS `Fiscal\_code`,

nome AS `Name`

FROM fornitori

WHERE nome LIKE var\_ricerca

ORDER BY codice\_fornitore DESC;

END$$

**OPC-01**

Il livello di isolamento scelto e' SERIALIZABLE alfine di presentare uno snapshot affidabile sull’insieme dei pacchi relativo all'ordineselezionato anche nel caso in cui esso sia ancora in lavorazionee quindi potrebbe mutare concorrentemente alla produzionedel report con eventuali inserimenti.

CREATE PROCEDURE `report\_pacchi` (IN var\_ordine INT)

BEGIN

DECLARE var\_numero\_pacco INT;

DECLARE done INT DEFAULT FALSE;

DECLARE cur CURSOR FOR

SELECT numero

FROM pacchi

WHERE ordine = var\_ordine;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

RESIGNAL;

END;

DROP TEMPORARY TABLE IF EXISTS dettagli\_pacchi;

CREATE TEMPORARY TABLE dettagli\_pacchi (

`Pack\_number` INT UNSIGNED,

`Species\_code` INT,

`Latin\_name` VARCHAR(64),

`Common\_name` VARCHAR(64),

`Quantity` INT UNSIGNED

);

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

START TRANSACTION;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO var\_numero\_pacco;

IF done THEN

LEAVE read\_loop;

END IF;

INSERT INTO dettagli\_pacchi

SELECT pacco AS `Pack\_number`,

codice AS `Species\_code`,

nome\_latino AS `Latin\_name`,

nome\_comune AS `Common\_name`,

quantita AS `Quantity`

FROM specie\_di\_piante INNER JOIN contenuto\_pacchi

ON codice = specie\_di\_pianta

WHERE ordine = var\_ordine AND

pacco = var\_numero\_pacco;

END LOOP;

CLOSE cur;

SELECT \*

FROM dettagli\_pacchi

ORDER BY `Pack\_number`;

DROP TEMPORARY TABLE dettagli\_pacchi;

COMMIT;

END$$

**OPC-02**

Il livello di isolamento scelto e' SERIALIZABLE sia per via dell'utilizzo della select max per ottenere il numero identificativo del pacco relativamente all’ordine a cui afferisce, sia per via dell’utilizzo della funzione aggregata sum utilizzata nel trigger BI di contenuto\_pacchi.

CREATE PROCEDURE `crea\_pacco` (

IN var\_ordine INT,

IN var\_specie INT,

IN var\_quantita INT,

OUT var\_numero INT

)

BEGIN

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1452 THEN

SIGNAL SQLSTATE '45027'

SET MESSAGE\_TEXT = "Either species or order does not exists";

ELSE

RESIGNAL;

END IF;

END;

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

START TRANSACTION;

SELECT MAX(numero) + 1

FROM pacchi

WHERE ordine = var\_ordine

INTO var\_numero;

IF var\_numero IS NULL THEN

SET var\_numero = 1;

END IF;

INSERT INTO pacchi (`ordine`, `numero`) VALUES (var\_ordine, var\_numero);

INSERT INTO contenuto\_pacchi(`ordine`, `pacco`, `specie\_di\_pianta`, `quantita`)

VALUES (var\_ordine, var\_numero, var\_specie, var\_quantita);

COMMIT;

END$$

**OPC-03**

Il livello di isolamento scelto e' SERIALIZABLE per via dell’utilizzo della funzione aggregata sum utilizzata nel trigger BI di contenuto\_pacchi.

CREATE PROCEDURE `aggiungi\_specie\_a\_pacco` (

IN var\_specie INT,

IN var\_ordine INT,

IN var\_pacco INT,

IN var\_quantita INT

)

BEGIN

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1452 THEN

SIGNAL SQLSTATE '45026'

SET MESSAGE\_TEXT = "Either species or pack does not exists";

ELSE

RESIGNAL;

END IF;

END;

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

START TRANSACTION;

INSERT INTO contenuto\_pacchi(`ordine`, `pacco`, `specie\_di\_pianta`, `quantita`)

VALUES (var\_ordine, var\_pacco, var\_specie, var\_quantita);

COMMIT;

END$$

**OPC-04**

Il livello di isolamento scelto è READ COMMITTED poiché è di interesse soltanto leggere dati consistenti in quanto i dati letti non vengono riutilizzati ma semplicemente restituiti all’utente.

CREATE PROCEDURE `visualizza\_piante\_rimanenti\_da\_impacchettare` (IN var\_ordine INT)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

RESIGNAL;

END;

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

START TRANSACTION;

SELECT specie\_di\_piante.codice AS `Species\_code`,

CONCAT(specie\_di\_piante.nome\_comune, " (",

specie\_di\_piante.nome\_latino, + ")") AS `Species\_name`,

appartenenza\_ordini.quantita AS `Quantity`,

CASE

WHEN SUM(contenuto\_pacchi.quantita) IS NULL THEN 0

ELSE SUM(contenuto\_pacchi.quantita)

END AS `Already\_processed`

FROM specie\_di\_piante LEFT JOIN contenuto\_pacchi

ON contenuto\_pacchi.specie\_di\_pianta = specie\_di\_piante.codice

INNER JOIN appartenenza\_ordini

ON appartenenza\_ordini.specie\_di\_pianta = specie\_di\_piante.codice

WHERE appartenenza\_ordini.ordine = var\_ordine AND

(contenuto\_pacchi.ordine = var\_ordine OR

contenuto\_pacchi.ordine IS NULL)

GROUP BY `Species\_code`, `Species\_name`, `Quantity`;

COMMIT;

END$$

**OPC-05**

Il livello di isolamento scelto è serializable per via dell’utilizzo della funzione verifica\_completamento che fa uso di funzioni aggregate.

CREATE PROCEDURE `visualizza\_stato\_ordine` (IN var\_ordine INT)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

RESIGNAL;

END;

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

START TRANSACTION;

SELECT `ordini`.`id` AS `Order\_ID`,

DATE(`ordini`.`data`) AS `Date`,

`ordini`.`indirizzo\_spedizione` AS `Shipping\_address`,

`ordini`.`contatto` AS `Chosen\_contact`,

CASE

WHEN verifica\_completamento(`id`) THEN "Completed"

WHEN aperto\_or\_finalizzato = 1 THEN "Already open"

ELSE "Not yet completed"

END AS `Status`,

CASE

WHEN `clienti`.`privato\_or\_rivendita` = 1

THEN "Not expected"

ELSE concat(`referenti`.`nome`, " ", `referenti`.`cognome`)

END AS `Referent`

FROM `ordini` INNER JOIN `clienti`

ON `ordini`.`cliente` = `clienti`.`codice`

LEFT JOIN `referenti`

ON `clienti`.`codice` = `referenti`.`rivendita`

WHERE `id` = var\_ordine;

COMMIT;

END$$

**NRG-01**

La transazione e' stata utilizzata in questo caso per effettuare un insieme di scritture in modalità "all or nothing" per cui non è necessario fissare un livello di isolamento (non avvengono letture).

CREATE PROCEDURE `registra\_privato` (

IN var\_codice CHAR(16),

IN var\_nome VARCHAR(32),

IN var\_res VARCHAR(64),

IN var\_fat VARCHAR(64),

IN var\_user VARCHAR(128),

IN var\_pass VARCHAR(128)

)

BEGIN

DECLARE var\_uuid CHAR(36);

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1062 THEN

SIGNAL SQLSTATE '45011'

SET MESSAGE\_TEXT = "User already exists";

ELSE

RESIGNAL;

END IF;

END;

START TRANSACTION;

SELECT UUID() INTO var\_uuid;

IF verifica\_password(var\_pass) = 1 THEN

SIGNAL SQLSTATE '45003'

SET MESSAGE\_TEXT = "Minimum password length is 8 characters";

ELSEIF verifica\_password(var\_pass) = 2 THEN

SIGNAL SQLSTATE '45004'

SET MESSAGE\_TEXT = "Password must not contains spaces";

ELSE

INSERT INTO utenti (`username`, `password`, `ruolo`, `uuid`)

VALUES (var\_user, SHA2(CONCAT(var\_pass, var\_uuid), 512),

"PCS", var\_uuid);

INSERT INTO clienti (`codice`, `nome`, `indirizzo\_residenza`,

`indirizzo\_fatturazione`, `privato\_or\_rivendita`)

VALUES (UPPER(var\_codice), var\_nome, var\_res, var\_fat, 1);

INSERT INTO utenze\_clienti (`utente`, `cliente`)

VALUES (var\_user, UPPER(var\_codice));

END IF;

COMMIT;

END$$

**NRG-02**

La transazione e' stata utilizzata in questo caso per effettuare un insieme di scritture in modalità "all or nothing" per cui non è necessario fissare un livello di isolamento (non avvengono letture).

CREATE PROCEDURE `registra\_rivendita` (

IN var\_codice CHAR(11),

IN var\_nome\_riv VARCHAR(32),

IN var\_res VARCHAR(64),

IN var\_fat VARCHAR(64),

IN var\_user VARCHAR(128),

IN var\_pass VARCHAR(128),

IN var\_nome\_ref VARCHAR(32),

IN var\_cognome\_ref VARCHAR(32)

)

BEGIN

DECLARE var\_uuid CHAR(36);

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1062 THEN

SIGNAL SQLSTATE '45011'

SET MESSAGE\_TEXT = "User already exists";

ELSE

RESIGNAL;

END IF;

END;

START TRANSACTION;

SELECT UUID() INTO var\_uuid;

IF verifica\_password(var\_pass) = 1 THEN

SIGNAL SQLSTATE '45003'

SET MESSAGE\_TEXT = "Minimum password length is 8 characters";

ELSEIF verifica\_password(var\_pass) = 2 THEN

SIGNAL SQLSTATE '45004'

SET MESSAGE\_TEXT = "Password must not contains spaces";

ELSE

INSERT INTO utenti (`username`, `password`, `ruolo`, `uuid`)

VALUES (var\_user, SHA2(CONCAT(var\_pass, var\_uuid), 512),

"RCS", var\_uuid);

INSERT INTO clienti (`codice`, `nome`, `indirizzo\_residenza`,

`indirizzo\_fatturazione`, `privato\_or\_rivendita`)

VALUES (var\_codice, var\_nome\_riv, var\_res, var\_fat, 0);

INSERT INTO utenze\_clienti (`utente`, `cliente`)

VALUES (var\_user, var\_codice);

INSERT INTO referenti (`rivendita`, `nome`, `cognome`)

VALUES (var\_codice, var\_nome\_ref, var\_cognome\_ref);

END IF;

COMMIT;

END$$

**NRG-03**

È stato necessario introdurre l'handler al fine di evitare il warning 1329: No data - zero rows fetched, selected, or processed

CREATE PROCEDURE `login` (

IN var\_username VARCHAR(128),

IN var\_password VARCHAR(128),

OUT var\_ruolo INT,

OUT var\_codice\_cliente VARCHAR(16)

)

BEGIN

DECLARE var\_ruolo\_enum ENUM("PCS", "RCS", "WHC",

"OPC", "MNG", "COS", "ERR");

DECLARE CONTINUE HANDLER FOR NOT FOUND SET var\_ruolo\_enum = "ERR";

SELECT ruolo

FROM utenti

WHERE `username` = var\_username and

`password` = SHA2(CONCAT(var\_password, `uuid`), 512)

INTO var\_ruolo\_enum;

IF var\_ruolo\_enum = "PCS" OR var\_ruolo\_enum = "RCS" THEN

SELECT cliente

FROM utenze\_clienti

WHERE utente = var\_username

INTO var\_codice\_cliente;

END IF;

CASE var\_ruolo\_enum

WHEN "PCS" THEN SET var\_ruolo = 0;

WHEN "RCS" THEN SET var\_ruolo = 1;

WHEN "WHC" THEN SET var\_ruolo = 2;

WHEN "OPC" THEN SET var\_ruolo = 3;

WHEN "MNG" THEN SET var\_ruolo = 4;

WHEN "COS" THEN SET var\_ruolo = 5;

ELSE SET var\_ruolo = 6;

END CASE;

END$$

**ALL-01**

Il livello di isolamento scelto per la transazione è READ COMMITTED per via del fatto che mediante tale stored procedure e' possibile modificare la password associata ad un'account e per effettuare un confronto corretto e' necessario leggere un dato gia' committato

CREATE PROCEDURE `modifica\_password` (

IN var\_username VARCHAR(32),

IN var\_vecchia\_password VARCHAR(128),

IN var\_nuova\_password VARCHAR(128)

)

BEGIN

DECLARE var\_vecchia\_password\_memorizzata VARCHAR(128);

DECLARE var\_uuid CHAR(36);

DECLARE CONTINUE HANDLER FOR NOT FOUND

SIGNAL SQLSTATE '45028' SET MESSAGE\_TEXT = "User not exists";

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

RESIGNAL;

END;

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

START TRANSACTION;

SELECT `password`, `uuid`

FROM `utenti`

WHERE `username` = var\_username

INTO var\_vecchia\_password\_memorizzata, var\_uuid;

IF var\_vecchia\_password\_memorizzata <>

SHA2(CONCAT(var\_vecchia\_password, var\_uuid), 512) THEN

SIGNAL SQLSTATE '45017'

SET MESSAGE\_TEXT = 'Old password inserted does not match

with the stored one';

END IF;

UPDATE `utenti`

SET `password` = SHA2(CONCAT(var\_nuova\_password, var\_uuid), 512)

WHERE `username` = var\_username;

COMMIT;

END$$

**COS-01**

La transazione e' stata utilizzata in questo caso per effettuare un insieme di scritture in modalità "all or nothing" per cui non è necessario fissare un livello di isolamento (non avvengono letture).

CREATE PROCEDURE `crea\_utenza\_dipendente` (

IN var\_username VARCHAR(128),

IN var\_password VARCHAR(128),

IN var\_ruolo CHAR(3)

)

BEGIN

DECLARE var\_uuid CHAR(36);

DECLARE var\_errno INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

GET DIAGNOSTICS CONDITION 1 var\_errno = MYSQL\_ERRNO;

IF var\_errno = 1062 THEN

SIGNAL SQLSTATE '45011'

SET MESSAGE\_TEXT = "User already exists";

ELSE

RESIGNAL;

END IF;

END;

START TRANSACTION;

IF var\_ruolo = "PCS" OR var\_ruolo = "RCS" THEN

SIGNAL SQLSTATE '45033'

SET MESSAGE\_TEXT = "Selected role is not an employee";

END IF;

SELECT UUID() INTO var\_uuid;

IF verifica\_password(var\_password) = 1 THEN

SIGNAL SQLSTATE '45003'

SET MESSAGE\_TEXT = "Minimum password length is 8 characters";

ELSEIF verifica\_password(var\_password) = 2 THEN

SIGNAL SQLSTATE '45004'

SET MESSAGE\_TEXT = "Password must not contains spaces";

ELSE

INSERT INTO utenti (`username`, `password`, `ruolo`, `uuid`)

VALUES (var\_username,

SHA2(CONCAT(var\_password, var\_uuid), 512), var\_ruolo, var\_uuid);

END IF;

COMMIT;

END$$

# Appendice: Implementazione

## Codice SQL per instanziare il database

**Creazione tabelle**

-- -----------------------------------------------------

-- Schema verdesrl

-- -----------------------------------------------------

DROP SCHEMA IF EXISTS `verdesrl` ;

-- -----------------------------------------------------

-- Schema verdesrl

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `verdesrl` ;

USE `verdesrl` ;

-- -----------------------------------------------------

-- Table `verdesrl`.`utenti`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `verdesrl`.`utenti` ;

CREATE TABLE IF NOT EXISTS `verdesrl`.`utenti` (

`username` VARCHAR(128) NOT NULL,

`password` CHAR(128) NOT NULL,

`ruolo` ENUM("PCS", "RCS", "WHC", "OPC", "MNG", "COS") NOT NULL,

`uuid` CHAR(36) NOT NULL,

PRIMARY KEY (`username`),

UNIQUE INDEX `uuid\_UNIQUE` (`uuid` ASC))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `verdesrl`.`specie\_di\_piante`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `verdesrl`.`specie\_di\_piante` ;

CREATE TABLE IF NOT EXISTS `verdesrl`.`specie\_di\_piante` (

`codice` INT NOT NULL AUTO\_INCREMENT,

`nome\_latino` VARCHAR(64) NOT NULL,

`nome\_comune` VARCHAR(64) NOT NULL,

`interni\_or\_esterni` TINYINT NOT NULL,

`verde\_or\_fiorita` TINYINT NOT NULL,

`esotica\_si\_no` TINYINT NOT NULL,

`giacenza` INT UNSIGNED NOT NULL DEFAULT 0,

PRIMARY KEY (`codice`),

UNIQUE INDEX `nome\_latino\_UNIQUE` (`nome\_latino` ASC),

FULLTEXT INDEX `nome\_comune\_FULLTEXT` (`nome\_comune`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `verdesrl`.`prezzi`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `verdesrl`.`prezzi` ;

CREATE TABLE IF NOT EXISTS `verdesrl`.`prezzi` (

`specie\_di\_pianta` INT NOT NULL,

`data\_adozione` DATETIME NOT NULL,

`valore` DECIMAL(7,2) UNSIGNED NOT NULL,

`data\_termine` DATETIME NULL,

PRIMARY KEY (`specie\_di\_pianta`, `data\_adozione`),

INDEX `fk\_prezzi\_specie\_di\_piante1\_idx` (`specie\_di\_pianta` ASC),

CONSTRAINT `fk\_prezzi\_specie\_di\_piante1`

FOREIGN KEY (`specie\_di\_pianta`)

REFERENCES `verdesrl`.`specie\_di\_piante` (`codice`)

ON DELETE CASCADE

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `verdesrl`.`contatti`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `verdesrl`.`contatti` ;

CREATE TABLE IF NOT EXISTS `verdesrl`.`contatti` (

`valore` VARCHAR(256) NOT NULL,

`tipo` ENUM("telefono", "cellulare", "email") NOT NULL,

PRIMARY KEY (`valore`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `verdesrl`.`clienti`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `verdesrl`.`clienti` ;

CREATE TABLE IF NOT EXISTS `verdesrl`.`clienti` (

`codice` VARCHAR(16) NOT NULL,

`nome` VARCHAR(32) NOT NULL,

`indirizzo\_residenza` VARCHAR(64) NOT NULL,

`indirizzo\_fatturazione` VARCHAR(64) NULL,

`privato\_or\_rivendita` TINYINT NOT NULL,

`contatto\_preferito` VARCHAR(256) NULL,

PRIMARY KEY (`codice`),

INDEX `fk\_clienti\_contatti1\_idx` (`contatto\_preferito` ASC),

CONSTRAINT `fk\_clienti\_contatti1`

FOREIGN KEY (`contatto\_preferito`)

REFERENCES `verdesrl`.`contatti` (`valore`)

ON DELETE SET NULL

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `verdesrl`.`referenti`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `verdesrl`.`referenti` ;

CREATE TABLE IF NOT EXISTS `verdesrl`.`referenti` (

`rivendita` VARCHAR(16) NOT NULL,

`nome` VARCHAR(32) NOT NULL,

`cognome` VARCHAR(32) NOT NULL,

`contatto\_preferito` VARCHAR(256) NULL,

INDEX `fk\_referenti\_contatti1\_idx` (`contatto\_preferito` ASC),

PRIMARY KEY (`rivendita`),

CONSTRAINT `fk\_referenti\_contatti1`

FOREIGN KEY (`contatto\_preferito`)

REFERENCES `verdesrl`.`contatti` (`valore`)

ON DELETE SET NULL

ON UPDATE NO ACTION,

CONSTRAINT `fk\_referenti\_clienti1`

FOREIGN KEY (`rivendita`)

REFERENCES `verdesrl`.`clienti` (`codice`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `verdesrl`.`ordini`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `verdesrl`.`ordini` ;

CREATE TABLE IF NOT EXISTS `verdesrl`.`ordini` (

`id` INT NOT NULL AUTO\_INCREMENT,

`cliente` VARCHAR(16) NOT NULL,

`data` DATETIME NOT NULL,

`aperto\_or\_finalizzato` TINYINT NOT NULL DEFAULT 1,

`indirizzo\_spedizione` VARCHAR(64) NOT NULL,

`contatto` VARCHAR(256) NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_ordini\_contatti1\_idx` (`contatto` ASC),

INDEX `fk\_ordini\_clienti1\_idx` (`cliente` ASC),

CONSTRAINT `fk\_ordini\_contatti1`

FOREIGN KEY (`contatto`)

REFERENCES `verdesrl`.`contatti` (`valore`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_ordini\_clienti1`

FOREIGN KEY (`cliente`)

REFERENCES `verdesrl`.`clienti` (`codice`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `verdesrl`.`pacchi`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `verdesrl`.`pacchi` ;

CREATE TABLE IF NOT EXISTS `verdesrl`.`pacchi` (

`ordine` INT NOT NULL,

`numero` INT UNSIGNED NOT NULL,

PRIMARY KEY (`ordine`, `numero`),

INDEX `fk\_pacchi\_ordini1\_idx` (`ordine` ASC),

CONSTRAINT `fk\_pacchi\_ordini1`

FOREIGN KEY (`ordine`)

REFERENCES `verdesrl`.`ordini` (`id`)

ON DELETE CASCADE

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `verdesrl`.`fornitori`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `verdesrl`.`fornitori` ;

CREATE TABLE IF NOT EXISTS `verdesrl`.`fornitori` (

`codice\_fornitore` INT NOT NULL AUTO\_INCREMENT,

`codice\_fiscale` CHAR(16) NOT NULL,

`nome` VARCHAR(32) NOT NULL,

PRIMARY KEY (`codice\_fornitore`),

UNIQUE INDEX `codice\_fiscale\_UNIQUE` (`codice\_fiscale` ASC))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `verdesrl`.`indirizzi\_fornitori`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `verdesrl`.`indirizzi\_fornitori` ;

CREATE TABLE IF NOT EXISTS `verdesrl`.`indirizzi\_fornitori` (

`indirizzo` VARCHAR(64) NOT NULL,

`fornitore` INT NOT NULL,

PRIMARY KEY (`indirizzo`),

INDEX `fk\_indirizzi\_fornitori\_fornitori1\_idx` (`fornitore` ASC),

CONSTRAINT `fk\_indirizzi\_fornitori\_fornitori1`

FOREIGN KEY (`fornitore`)

REFERENCES `verdesrl`.`fornitori` (`codice\_fornitore`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `verdesrl`.`richieste\_di\_forniture`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `verdesrl`.`richieste\_di\_forniture` ;

CREATE TABLE IF NOT EXISTS `verdesrl`.`richieste\_di\_forniture` (

`data` DATETIME NOT NULL,

`fornitore\_scelto` INT NOT NULL,

`specie\_richiesta` INT NOT NULL,

`quantita` INT UNSIGNED NOT NULL DEFAULT 0,

`pendente\_si\_no` TINYINT NOT NULL DEFAULT 1,

PRIMARY KEY (`data`, `fornitore\_scelto`, `specie\_richiesta`),

INDEX `fk\_richieste\_di\_forniture\_specie\_di\_piante1\_idx` (`specie\_richiesta` ASC),

INDEX `fk\_richieste\_di\_forniture\_fornitori1\_idx` (`fornitore\_scelto` ASC),

CONSTRAINT `fk\_richieste\_di\_forniture\_specie\_di\_piante1`

FOREIGN KEY (`specie\_richiesta`)

REFERENCES `verdesrl`.`specie\_di\_piante` (`codice`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_richieste\_di\_forniture\_fornitori1`

FOREIGN KEY (`fornitore\_scelto`)

REFERENCES `verdesrl`.`fornitori` (`codice\_fornitore`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `verdesrl`.`colorazioni`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `verdesrl`.`colorazioni` ;

CREATE TABLE IF NOT EXISTS `verdesrl`.`colorazioni` (

`specie\_fiorita` INT NOT NULL,

`colore` VARCHAR(32) NOT NULL,

PRIMARY KEY (`specie\_fiorita`, `colore`),

CONSTRAINT `fk\_colorazioni\_specie\_di\_piante1`

FOREIGN KEY (`specie\_fiorita`)

REFERENCES `verdesrl`.`specie\_di\_piante` (`codice`)

ON DELETE CASCADE

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `verdesrl`.`contatti\_clienti`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `verdesrl`.`contatti\_clienti` ;

CREATE TABLE IF NOT EXISTS `verdesrl`.`contatti\_clienti` (

`contatto` VARCHAR(256) NOT NULL,

`cliente` VARCHAR(16) NOT NULL,

PRIMARY KEY (`contatto`),

INDEX `fk\_contatti\_clienti\_clienti1\_idx` (`cliente` ASC),

CONSTRAINT `fk\_contatti\_clienti\_contatti1`

FOREIGN KEY (`contatto`)

REFERENCES `verdesrl`.`contatti` (`valore`)

ON DELETE CASCADE

ON UPDATE NO ACTION,

CONSTRAINT `fk\_contatti\_clienti\_clienti1`

FOREIGN KEY (`cliente`)

REFERENCES `verdesrl`.`clienti` (`codice`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `verdesrl`.`contatti\_referenti`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `verdesrl`.`contatti\_referenti` ;

CREATE TABLE IF NOT EXISTS `verdesrl`.`contatti\_referenti` (

`contatto` VARCHAR(256) NOT NULL,

`referente` VARCHAR(16) NOT NULL,

PRIMARY KEY (`contatto`),

INDEX `fk\_contatti\_referenti\_referenti1\_idx` (`referente` ASC),

CONSTRAINT `fk\_contatti\_referenti\_contatti1`

FOREIGN KEY (`contatto`)

REFERENCES `verdesrl`.`contatti` (`valore`)

ON DELETE CASCADE

ON UPDATE NO ACTION,

CONSTRAINT `fk\_contatti\_referenti\_referenti1`

FOREIGN KEY (`referente`)

REFERENCES `verdesrl`.`referenti` (`rivendita`)

ON DELETE CASCADE

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `verdesrl`.`contenuto\_pacchi`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `verdesrl`.`contenuto\_pacchi` ;

CREATE TABLE IF NOT EXISTS `verdesrl`.`contenuto\_pacchi` (

`ordine` INT NOT NULL,

`pacco` INT UNSIGNED NOT NULL,

`specie\_di\_pianta` INT NOT NULL,

`quantita` INT UNSIGNED NOT NULL DEFAULT 0,

PRIMARY KEY (`ordine`, `pacco`, `specie\_di\_pianta`),

INDEX `fk\_contenuto\_pacchi\_specie\_di\_piante1\_idx` (`specie\_di\_pianta` ASC),

INDEX `fk\_contenuto\_pacchi\_pacchi1\_idx` (`ordine` ASC, `pacco` ASC),

CONSTRAINT `fk\_contenuto\_pacchi\_specie\_di\_piante1`

FOREIGN KEY (`specie\_di\_pianta`)

REFERENCES `verdesrl`.`specie\_di\_piante` (`codice`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_contenuto\_pacchi\_pacchi1`

FOREIGN KEY (`ordine` , `pacco`)

REFERENCES `verdesrl`.`pacchi` (`ordine` , `numero`)

ON DELETE CASCADE

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `verdesrl`.`disponibilita\_forniture`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `verdesrl`.`disponibilita\_forniture` ;

CREATE TABLE IF NOT EXISTS `verdesrl`.`disponibilita\_forniture` (

`specie\_di\_pianta` INT NOT NULL,

`fornitore` INT NOT NULL,

PRIMARY KEY (`specie\_di\_pianta`, `fornitore`),

INDEX `fk\_disponibilta\_forniture\_specie\_di\_piante1\_idx` (`specie\_di\_pianta` ASC),

INDEX `fk\_disponibilta\_forniture\_fornitori1\_idx` (`fornitore` ASC),

CONSTRAINT `fk\_disponibilta\_forniture\_specie\_di\_piante1`

FOREIGN KEY (`specie\_di\_pianta`)

REFERENCES `verdesrl`.`specie\_di\_piante` (`codice`)

ON DELETE CASCADE

ON UPDATE NO ACTION,

CONSTRAINT `fk\_disponibilta\_forniture\_fornitori1`

FOREIGN KEY (`fornitore`)

REFERENCES `verdesrl`.`fornitori` (`codice\_fornitore`)

ON DELETE CASCADE

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `verdesrl`.`utenze\_clienti`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `verdesrl`.`utenze\_clienti` ;

CREATE TABLE IF NOT EXISTS `verdesrl`.`utenze\_clienti` (

`cliente` VARCHAR(128) NOT NULL,

`utente` VARCHAR(32) NOT NULL,

PRIMARY KEY (`cliente`),

INDEX `fk\_utenze\_clienti\_utenti1\_idx` (`utente` ASC),

CONSTRAINT `fk\_utenze\_clienti\_clienti1`

FOREIGN KEY (`cliente`)

REFERENCES `verdesrl`.`clienti` (`codice`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_utenze\_clienti\_utenti1`

FOREIGN KEY (`utente`)

REFERENCES `verdesrl`.`utenti` (`username`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `verdesrl`.`appartenenza\_ordini`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `verdesrl`.`appartenenza\_ordini` ;

CREATE TABLE IF NOT EXISTS `verdesrl`.`appartenenza\_ordini` (

`ordine` INT NOT NULL,

`specie\_di\_pianta` INT NOT NULL,

`quantita` INT UNSIGNED NOT NULL DEFAULT 0,

PRIMARY KEY (`ordine`, `specie\_di\_pianta`),

INDEX `fk\_appartenenza\_ordini\_specie\_di\_piante1\_idx` (`specie\_di\_pianta` ASC),

INDEX `fk\_appartenenza\_ordini\_ordini1\_idx` (`ordine` ASC),

CONSTRAINT `fk\_appartenenza\_ordini\_specie\_di\_piante1`

FOREIGN KEY (`specie\_di\_pianta`)

REFERENCES `verdesrl`.`specie\_di\_piante` (`codice`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_appartenenza\_ordini\_ordini1`

FOREIGN KEY (`ordine`)

REFERENCES `verdesrl`.`ordini` (`id`)

ON DELETE CASCADE

ON UPDATE NO ACTION)

ENGINE = InnoDB;

**Utenti e privilegi**

DROP USER IF EXISTS non\_registrato;

SET SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

CREATE USER 'non\_registrato' IDENTIFIED BY 'verdesrl';

GRANT EXECUTE ON procedure `verdesrl`.`login` TO 'non\_registrato';

GRANT EXECUTE ON procedure `verdesrl`.`registra\_privato` TO 'non\_registrato';

GRANT EXECUTE ON procedure `verdesrl`.`registra\_rivendita` TO 'non\_registrato';

SET SQL\_MODE = '';

DROP USER IF EXISTS cliente\_privato;

SET SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

CREATE USER 'cliente\_privato' IDENTIFIED BY 'verdesrl';

GRANT EXECUTE ON procedure `verdesrl`.`modifica\_password` TO 'cliente\_privato';

GRANT EXECUTE ON procedure `verdesrl`.`crea\_ordine` TO 'cliente\_privato';

GRANT EXECUTE ON procedure `verdesrl`.`rimuovi\_specie\_da\_ordine` TO 'cliente\_privato';

GRANT EXECUTE ON procedure `verdesrl`.`modifica\_ordine` TO 'cliente\_privato';

GRANT EXECUTE ON procedure `verdesrl`.`aggiungi\_specie\_ad\_ordine\_esistente` TO 'cliente\_privato';

GRANT EXECUTE ON procedure `verdesrl`.`finalizza\_ordine` TO 'cliente\_privato';

GRANT EXECUTE ON procedure `verdesrl`.`report\_ordine` TO 'cliente\_privato';

GRANT EXECUTE ON procedure `verdesrl`.`modifica\_residenza` TO 'cliente\_privato';

GRANT EXECUTE ON procedure `verdesrl`.`modifica\_fatturazione` TO 'cliente\_privato';

GRANT EXECUTE ON procedure `verdesrl`.`aggiungi\_contatto\_cliente` TO 'cliente\_privato';

GRANT EXECUTE ON procedure `verdesrl`.`rimuovi\_contatto\_cliente` TO 'cliente\_privato';

GRANT EXECUTE ON procedure `verdesrl`.`modifica\_contatto\_preferito\_cliente` TO 'cliente\_privato';

GRANT EXECUTE ON procedure `verdesrl`.`visualizza\_ordini\_cliente` TO 'cliente\_privato';

GRANT EXECUTE ON procedure `verdesrl`.`visualizza\_contatti\_cliente` TO 'cliente\_privato';

GRANT EXECUTE ON procedure `verdesrl`.`visualizza\_dettagli\_specie` TO 'cliente\_privato';

GRANT EXECUTE ON procedure `verdesrl`.`visualizza\_specie\_appartenenti\_ad\_ordine` TO 'cliente\_privato';

SET SQL\_MODE = '';

DROP USER IF EXISTS cliente\_rivendita;

SET SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

CREATE USER 'cliente\_rivendita' IDENTIFIED BY 'verdesrl';

GRANT EXECUTE ON procedure `verdesrl`.`modifica\_password` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`crea\_ordine` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`rimuovi\_specie\_da\_ordine` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`modifica\_ordine` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`aggiungi\_specie\_ad\_ordine\_esistente` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`finalizza\_ordine` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`report\_ordine` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`modifica\_residenza` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`modifica\_fatturazione` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`aggiungi\_contatto\_cliente` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`rimuovi\_contatto\_cliente` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`modifica\_contatto\_preferito\_cliente` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`aggiungi\_contatto\_referente` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`rimuovi\_contatto\_referente` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`modifica\_contatto\_preferito\_referente` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`visualizza\_ordini\_cliente` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`visualizza\_contatti\_cliente` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`visualizza\_contatti\_referente` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`visualizza\_dettagli\_specie` TO 'cliente\_rivendita';

GRANT EXECUTE ON procedure `verdesrl`.`visualizza\_specie\_appartenenti\_ad\_ordine` TO 'cliente\_rivendita';

SET SQL\_MODE = '';

DROP USER IF EXISTS addetto\_diparimento\_magazzino;

SET SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

CREATE USER 'addetto\_diparimento\_magazzino' IDENTIFIED BY 'verdesrl';

GRANT EXECUTE ON procedure `verdesrl`.`modifica\_password` TO 'addetto\_diparimento\_magazzino';

GRANT EXECUTE ON procedure `verdesrl`.`inserisci\_richiesta\_fornitura` TO 'addetto\_diparimento\_magazzino';

GRANT EXECUTE ON procedure `verdesrl`.`inserisci\_fornitore` TO 'addetto\_diparimento\_magazzino';

GRANT EXECUTE ON procedure `verdesrl`.`aggiungi\_disponibilita\_fornitura` TO 'addetto\_diparimento\_magazzino';

GRANT EXECUTE ON procedure `verdesrl`.`report\_giacenza` TO 'addetto\_diparimento\_magazzino';

GRANT EXECUTE ON procedure `verdesrl`.`visualizza\_dettagli\_giacenza` TO 'addetto\_diparimento\_magazzino';

GRANT EXECUTE ON procedure `verdesrl`.`visualizza\_dettagli\_specie` TO 'addetto\_diparimento\_magazzino';

GRANT EXECUTE ON procedure `verdesrl`.`visualizza\_fornitori` TO 'addetto\_diparimento\_magazzino';

GRANT EXECUTE ON procedure `verdesrl`.`visualizza\_specie\_disponibili` TO 'addetto\_diparimento\_magazzino';

GRANT EXECUTE ON procedure `verdesrl`.`aggiungi\_indirizzo\_fornitore` TO 'addetto\_diparimento\_magazzino';

SET SQL\_MODE = '';

DROP USER IF EXISTS operatore\_pacchi;

SET SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

CREATE USER 'operatore\_pacchi' IDENTIFIED BY 'verdesrl';

GRANT EXECUTE ON procedure `verdesrl`.`modifica\_password` TO 'operatore\_pacchi';

GRANT EXECUTE ON procedure `verdesrl`.`aggiungi\_specie\_a\_pacco` TO 'operatore\_pacchi';

GRANT EXECUTE ON procedure `verdesrl`.`report\_pacchi` TO 'operatore\_pacchi';

GRANT EXECUTE ON procedure `verdesrl`.`visualizza\_stato\_ordine` TO 'operatore\_pacchi';

GRANT EXECUTE ON procedure `verdesrl`.`visualizza\_piante\_rimanenti\_da\_impacchettare` TO 'operatore\_pacchi';

GRANT EXECUTE ON procedure `verdesrl`.`crea\_pacco` TO 'operatore\_pacchi';

SET SQL\_MODE = '';

DROP USER IF EXISTS manager;

SET SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

CREATE USER 'manager' IDENTIFIED BY 'verdesrl';

GRANT EXECUTE ON procedure `verdesrl`.`modifica\_password` TO 'manager';

GRANT EXECUTE ON procedure `verdesrl`.`modifica\_prezzo` TO 'manager';

GRANT EXECUTE ON procedure `verdesrl`.`inserisci\_nuova\_specie` TO 'manager';

GRANT EXECUTE ON procedure `verdesrl`.`rimuovi\_specie\_di\_pianta` TO 'manager';

GRANT EXECUTE ON procedure `verdesrl`.`aggiungi\_colorazione` TO 'manager';

GRANT EXECUTE ON procedure `verdesrl`.`rimuovi\_colorazione` TO 'manager';

GRANT EXECUTE ON procedure `verdesrl`.`report\_specie` TO 'manager';

GRANT EXECUTE ON procedure `verdesrl`.`visualizza\_colorazioni` TO 'manager';

GRANT EXECUTE ON procedure `verdesrl`.`visualizza\_dettagli\_specie` TO 'manager';

SET SQL\_MODE = '';

DROP USER IF EXISTS capo\_personale;

SET SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

CREATE USER 'capo\_personale' IDENTIFIED BY 'verdesrl';

GRANT EXECUTE ON procedure `verdesrl`.`modifica\_password` TO 'capo\_personale';

GRANT EXECUTE ON procedure `verdesrl`.`crea\_utenza\_dipendente` TO 'capo\_personale';

**Insert predefinite**

In fase di configurazione è previsto l’inserimento di un record all’interno della tabella utenti per il login dell’amministratore del sistema. Di seguito sono portati dettagli e codice:

username: admin

password: admin

ruolo: COS (Chief Of Staff)

START TRANSACTION;

USE `verdesrl`;

INSERT INTO `verdesrl`.`utenti` (`username`, `password`, `ruolo`, `uuid`) VALUES ('admin', '7603e55f21a11a3c9e71a00110dd97cc1552d7954bf90685eaa91c98bbfac26e24d4e9af41a122a212ccd423a7e305820cb57c7cfaa88409c78ff832859532b3', 'COS', '058fa777-84c0-11ea-8c72-d0039b002ee5');

COMMIT;

**Funzioni**

CREATE FUNCTION `verifica\_proprietario`(id\_ordine INT, codice\_cliente VARCHAR(16))

RETURNS BOOL

DETERMINISTIC

BEGIN

DECLARE var\_cliente\_ordine VARCHAR(16);

SELECT cliente

FROM ordini

WHERE id = id\_ordine

INTO var\_cliente\_ordine;

RETURN (var\_cliente\_ordine = codice\_cliente);

END $$

CREATE FUNCTION `verifica\_password`(var\_password VARCHAR(128))

RETURNS TINYINT

DETERMINISTIC

BEGIN

IF LENGTH(var\_password) < 8 THEN

RETURN 1;

END IF;

IF INSTR(var\_password, " ") > 0 THEN

RETURN 2;

END IF;

RETURN 0;

END $$

CREATE FUNCTION `verifica\_completamento`(var\_ordine INT)

RETURNS BOOL

DETERMINISTIC

BEGIN

DECLARE var\_quantita\_in\_pacchi INT UNSIGNED;

DECLARE var\_quantita\_in\_ordine INT UNSIGNED;

SELECT SUM(quantita)

FROM appartenenza\_ordini

WHERE ordine = var\_ordine

INTO var\_quantita\_in\_ordine;

SELECT SUM(quantita)

FROM contenuto\_pacchi

WHERE ordine = var\_ordine

INTO var\_quantita\_in\_pacchi;

RETURN var\_quantita\_in\_pacchi = var\_quantita\_in\_ordine;

END $$

## Codice del Front-End

FILE: chiefofstaff.c

#include <stdio.h>

#include <string.h>

#include <stdbool.h>

#include <mysql.h>

#include <stdlib.h>

#include "defines.h"

static char curr\_user[BUFFSIZE\_L];

static bool attempt\_add\_employee\_account(char \*username, char \*password, char \*role)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[3];

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_username VARCHAR(128)

param[0].buffer = username;

param[0].buffer\_length = strlen(username);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_username VARCHAR(128)

param[1].buffer = password;

param[1].buffer\_length = strlen(password);

param[2].buffer\_type = MYSQL\_TYPE\_STRING; // IN var\_username VARCHAR(128)

param[2].buffer = role;

param[2].buffer\_length = strlen(role);

if (!exec\_sp(&stmt, param, "call crea\_utenza\_dipendente(?, ?, ?)"))

return false;

mysql\_stmt\_close(stmt);

return true;

}

static void add\_employee\_account(void)

{

char username[BUFFSIZE\_L];

char password[BUFFSIZE\_L];

char password\_check[BUFFSIZE\_L];

char role[4];

char choice;

memset(username, 0, sizeof(username));

memset(password, 0, sizeof(password));

memset(password\_check, 0, sizeof(password\_check));

init\_screen(false);

printf("Insert username: ");

get\_input(BUFFSIZE\_L, username, false, true);

retype\_pass:

printf("Insert password: ");

get\_input(BUFFSIZE\_L, password, true, true);

printf("Retype password: ");

get\_input(BUFFSIZE\_L, password\_check, true, true);

if (strcmp(password, password\_check) != 0) {

printf("Mismatch password, please retry!\n");

goto retype\_pass;

}

printf("\nWhat will be her/his role in the company?\n");

printf("1) Warehouse clerk\n");

printf("2) Order processor\n");

printf("3) Manager\n");

choice = multi\_choice("Pick an option", "123", 3);

switch (choice) {

case '1': snprintf(role, 4, "WHC"); break;

case '2': snprintf(role, 4, "OPC"); break;

case '3': snprintf(role, 4, "MNG"); break;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

putchar('\n');

if (attempt\_add\_employee\_account(username, password, role))

printf("Employee account for %s [%s] was succesfylly created\n", username, role);

else

printf("Operation failed\n");

printf("Press enter key to get back to menu ...\n");

getchar();

}

void run\_as\_chief\_of\_staff(char \*username)

{

struct configuration cnf;

char choice;

memset(&cnf, 0, sizeof(cnf));

memset(curr\_user, 0, sizeof(curr\_user));

strncpy(curr\_user, username, BUFFSIZE\_L);

if (parse\_config("config/cos.user", &cnf, "=")) {

fprintf(stderr, "Invalid configuration file selected (COS)\n");

exit(EXIT\_FAILURE);

}

if(mysql\_change\_user(conn, cnf.username, cnf.password, cnf.database)) {

fprintf(stderr, "Unable to switch privileges\n");

exit(EXIT\_FAILURE);

}

while (true) {

init\_screen(true);

printf("Welcome %s\n\n", curr\_user);

printf("\*\*\* What do you wanna do? \*\*\*\n\n");

printf("a) Add an employee account\n");

printf("p) Change password\n");

printf("q) Quit\n");

choice = multi\_choice("Pick an option", "apq", 3);

switch (choice) {

case 'a': add\_employee\_account(); break;

case 'p': change\_password(curr\_user); break;

case 'q': printf("Bye bye!\n\n\n"); return;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

}

}

FILE: customer.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

#include <unistd.h>

#include "defines.h"

struct customer\_info {

char username[BUFFSIZE\_L];

char code[BUFFSIZE\_XS];

bool is\_private;

};

struct create\_order\_sp\_params {

char shipping\_address[BUFFSIZE\_M];

char contact[BUFFSIZE\_XL];

unsigned int species;

unsigned int quantity;

};

struct order\_info {

unsigned int id;

char date[BUFFSIZE\_S];

char shipping\_address[BUFFSIZE\_M];

char chosen\_contact[BUFFSIZE\_XL];

char status[BUFFSIZE\_XS];

/\*

\* nel report viene presentato come <nome> <cognome>

\* -> 32 + 1 + 32 + 1 = 66

\*/

char referent[2 \* BUFFSIZE\_S];

};

static struct customer\_info curr\_customer;

static bool attempt\_search\_species\_belonging\_to\_order(unsigned int order\_id)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[1];

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_ordine INT

param[0].buffer = &order\_id;

param[0].buffer\_length = sizeof(order\_id);

if (!exec\_sp(&stmt, param, "call visualizza\_specie\_appartenenti\_ad\_ordine(?)"))

return false;

if (!dump\_result\_set(stmt, "Species belonging to selected order:", LEADING\_ZERO\_BITMASK\_IDX\_0)) {

CLOSE\_AND\_RETURN(false, stmt);

}

mysql\_stmt\_close(stmt);

return true;

}

static void search\_species(void)

{

char name[BUFFSIZE\_M];

memset(name, 0, sizeof(name));

init\_screen(false);

printf("\*\*\* Search species by name \*\*\*\n");

printf("Insert the name to filter on (default all): ");

get\_input(BUFFSIZE\_M, name, false, false);

putchar('\n');

if (!attempt\_search\_species(false, name))

printf("Operation failed\n");

printf("\nPress enter key to get back to menu ...\n");

getchar();

}

static bool attempt\_report\_orders\_short(bool only\_open)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[2];

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_cliente VARCHAR(16)

param[0].buffer = curr\_customer.code;

param[0].buffer\_length = strlen(curr\_customer.code);

param[1].buffer\_type = MYSQL\_TYPE\_TINY; // IN var\_status TINYINT

param[1].buffer = &(only\_open);

param[1].buffer\_length = sizeof(only\_open);

if (!exec\_sp(&stmt, param, "call visualizza\_ordini\_cliente(?, ?)"))

return false;

if (!dump\_result\_set(stmt, "Open orders:", LEADING\_ZERO\_BITMASK\_IDX\_0)) {

CLOSE\_AND\_RETURN(false, stmt);

}

mysql\_stmt\_close(stmt);

return true;

}

static unsigned int attempt\_open\_order(struct create\_order\_sp\_params \*input)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[6];

unsigned int id;

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_cliente VARCHAR(16)

param[0].buffer = curr\_customer.code;

param[0].buffer\_length = strlen(curr\_customer.code);

if (strlen(input->shipping\_address) > 0) {

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_ind\_sped VARCHAR(64)

param[1].buffer = input->shipping\_address;

param[1].buffer\_length = strlen(input->shipping\_address);

} else {

param[1].buffer\_type = MYSQL\_TYPE\_NULL; // IN var\_ind\_sped VARCHAR(64)

param[1].buffer = NULL;

param[1].buffer\_length = 0;

}

if (strlen(input->contact) > 0) {

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_contatto VARCHAR(64)

param[2].buffer = input->contact;

param[2].buffer\_length = strlen(input->contact);

} else {

param[2].buffer\_type = MYSQL\_TYPE\_NULL; // IN var\_contatto VARCHAR(64)

param[2].buffer = NULL;

param[2].buffer\_length = 0;

}

param[3].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_specie INT

param[3].buffer = &(input->species);

param[3].buffer\_length = sizeof(input->species);

param[4].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_quantita INT

param[4].buffer = &(input->quantity);

param[4].buffer\_length = sizeof(input->quantity);

param[5].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_id INT

param[5].buffer = &id;

param[5].buffer\_length = sizeof(id);

if (!exec\_sp(&stmt, param, "call crea\_ordine(?, ?, ?, ?, ?, ?)"))

return 0;

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_id INT

param[0].buffer = &id;

param[0].buffer\_length = sizeof(id);

if (!fetch\_res\_sp(stmt, param))

return 0;

mysql\_stmt\_close(stmt);

return id;

}

static void open\_order(void)

{

struct create\_order\_sp\_params params;

char buffer\_for\_integer[BUFFSIZE\_XS];

unsigned int order\_id;

char spec\_name[BUFFSIZE\_M];

memset(&params, 0, sizeof(params));

memset(buffer\_for\_integer, 0, sizeof(buffer\_for\_integer));

memset(spec\_name, 0, sizeof(spec\_name));

init\_screen(false);

printf("\*\*\* Open a new order \*\*\*\n");

printf("Customer code....................................................: %s\n", curr\_customer.code);

printf("Insert shipping address (default residential address)............: ");

get\_input(BUFFSIZE\_M, params.shipping\_address, false, false);

printf("Insert contact (default favourite one)...........................: ");

get\_input(BUFFSIZE\_XL, params.contact, false, false);

if (ask\_for\_tips("Do you wanna search species by name to find the right code", 0)) {

printf("\nInsert the name to filter on (default all).......................: ");

get\_input(BUFFSIZE\_M, spec\_name, false, false);

if (!attempt\_search\_species(false, spec\_name))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert species code..............................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

params.species = strtol(buffer\_for\_integer, NULL, 10);

printf("Insert relative quantity.........................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

params.quantity = strtol(buffer\_for\_integer, NULL, 10);

putchar('\n');

order\_id = attempt\_open\_order(&params);

if (order\_id > 0)

printf("New order opened (ID: %010u)\n", order\_id);

else

printf("No order has opened\n");

printf("Press enter key to get back to menu ...\n");

getchar();

}

static bool attempt\_to\_add\_spec\_to\_order(unsigned int order\_id, unsigned int species\_code, unsigned int quantity)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[4];

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_cliente VARCHAR(16)

param[0].buffer = curr\_customer.code;

param[0].buffer\_length = strlen(curr\_customer.code);

param[1].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_specie INT

param[1].buffer = &species\_code;

param[1].buffer\_length = sizeof(species\_code);

param[2].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_ordine INT

param[2].buffer = &order\_id;

param[2].buffer\_length = sizeof(order\_id);

param[3].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_quantita INT

param[3].buffer = &quantity;

param[3].buffer\_length = sizeof(quantity);

if (!exec\_sp(&stmt, param, "call aggiungi\_specie\_ad\_ordine\_esistente(?, ?, ?, ?)"))

return false;

mysql\_stmt\_close(stmt);

return true;

}

static int attempt\_to\_modify\_order(unsigned int order\_id, unsigned int species\_code, unsigned int quantity)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[5];

int affected\_rows;

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_cliente VARCHAR(16)

param[0].buffer = curr\_customer.code;

param[0].buffer\_length = strlen(curr\_customer.code);

param[1].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_specie INT

param[1].buffer = &species\_code;

param[1].buffer\_length = sizeof(species\_code);

param[2].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_ordine INT

param[2].buffer = &order\_id;

param[2].buffer\_length = sizeof(order\_id);

param[3].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_quantita INT

param[3].buffer = &quantity;

param[3].buffer\_length = sizeof(quantity);

param[4].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_aggiornamento\_eff INT

param[4].buffer = &affected\_rows;

param[4].buffer\_length = sizeof(affected\_rows);

if (!exec\_sp(&stmt, param, "call modifica\_ordine(?, ?, ?, ?, ?)"))

return -1;

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_aggiornamento\_eff INT

param[0].buffer = &affected\_rows;

param[0].buffer\_length = sizeof(affected\_rows);

if (!fetch\_res\_sp(stmt, param))

return -1;

mysql\_stmt\_close(stmt);

return affected\_rows;

}

static void exec\_op\_on\_order(bool is\_add)

{

char buffer\_for\_integer[BUFFSIZE\_XS];

char spec\_name[BUFFSIZE\_M];

unsigned int order\_id;

unsigned int species\_code;

unsigned int quantity;

int ret;

memset(buffer\_for\_integer, 0, sizeof(buffer\_for\_integer));

memset(spec\_name, 0, sizeof(spec\_name));

init\_screen(false);

if (is\_add)

printf("\*\*\* Add a species to already opened order \*\*\*\n");

else

printf("\*\*\* Change the number of plants belonging to a species in an order \*\*\*\n");

printf("Customer code....................................................%s: %s\n",

(is\_add) ? "" : "....", curr\_customer.code);

if (ask\_for\_tips("Do you wanna see a report of your open orders", (is\_add) ? 13 : 17)) {

if (!attempt\_report\_orders\_short(true))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert order id..................................................%s: ", (is\_add) ? "" : "....");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

order\_id = strtol(buffer\_for\_integer, NULL, 10);

if (is\_add) {

if (ask\_for\_tips("Do you wanna search species by name to find the right code", 0)) {

printf("\nInsert the name to filter on (default all).......................: ");

get\_input(BUFFSIZE\_M, spec\_name, false, false);

if (!attempt\_search\_species(false, spec\_name))

printf("Operation failed\n");

putchar('\n');

}

} else {

if (ask\_for\_tips("Do you wanna see a list of species belonging to selected order", 0)) {

if (!attempt\_search\_species\_belonging\_to\_order(order\_id))

printf("Operation failed\n");

putchar('\n');

}

}

printf("Insert species code..............................................%s: ", (is\_add) ? "" : "....");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

species\_code = strtol(buffer\_for\_integer, NULL, 10);

printf("Insert relative quantity.........................................%s: ", (is\_add) ? "" : "....");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

quantity = strtol(buffer\_for\_integer, NULL, 10);

putchar('\n');

if (is\_add) {

if (attempt\_to\_add\_spec\_to\_order(order\_id, species\_code, quantity))

printf("Species %u succesfully added to your order (ID %010u)\n", species\_code, order\_id);

else

printf("Operation failed\n");

} else {

ret = attempt\_to\_modify\_order(order\_id, species\_code, quantity);

if (ret == 0)

printf("Nothing has changed (species %u not in order [ID %010u])\n", species\_code, order\_id);

else if (ret > 0)

printf("Species %u succesfully updated in your order (ID %010u)\n", species\_code, order\_id);

else

printf("Operation failed\n");

}

printf("Press enter key to get back to menu ...\n");

getchar();

}

static int attempt\_to\_remove\_spec\_from\_order(unsigned int order\_id, unsigned int species\_code, int \*order\_status)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[5];

int affected\_rows;

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_cliente VARCHAR(16)

param[0].buffer = curr\_customer.code;

param[0].buffer\_length = strlen(curr\_customer.code);

param[1].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_specie INT

param[1].buffer = &species\_code;

param[1].buffer\_length = sizeof(species\_code);

param[2].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_ordine INT

param[2].buffer = &order\_id;

param[2].buffer\_length = sizeof(order\_id);

param[3].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_ordine\_eliminato\_si\_no INT

param[3].buffer = order\_status;

param[3].buffer\_length = sizeof(\*order\_status);

param[4].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_eliminazione\_eff INT

param[4].buffer = &affected\_rows;

param[4].buffer\_length = sizeof(affected\_rows);

if (!exec\_sp(&stmt, param, "call rimuovi\_specie\_da\_ordine(?, ?, ?, ?, ?)"))

return -1;

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_ordine\_eliminato\_si\_no INT

param[0].buffer = order\_status;

param[0].buffer\_length = sizeof(\*order\_status);

param[1].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_eliminazione\_eff INT

param[1].buffer = &affected\_rows;

param[1].buffer\_length = sizeof(affected\_rows);

if (!fetch\_res\_sp(stmt, param))

return -1;

mysql\_stmt\_close(stmt);

return affected\_rows;

}

static void remove\_spec\_from\_order(void)

{

char buffer\_for\_integer[BUFFSIZE\_XS];

unsigned int order\_id;

unsigned int species\_code;

int order\_status;

int ret;

memset(buffer\_for\_integer, 0, sizeof(buffer\_for\_integer));

init\_screen(false);

printf("\*\*\* Remove a species from an order not closed yet \*\*\*\n");

printf("Customer code........................................................: %s\n",

curr\_customer.code);

if (ask\_for\_tips("Do you wanna see a report of your open orders", 17)) {

if (!attempt\_report\_orders\_short(true))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert order id......................................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

order\_id = strtol(buffer\_for\_integer, NULL, 10);

if (ask\_for\_tips("Do you wanna see a list of species belonging to selected order", 0)) {

if (!attempt\_search\_species\_belonging\_to\_order(order\_id))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert species code..................................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

species\_code = strtol(buffer\_for\_integer, NULL, 10);

putchar('\n');

ret = attempt\_to\_remove\_spec\_from\_order(order\_id, species\_code, &order\_status);

if (ret > 0) {

printf("Species %u succesfully deleted from your order (ID %010u)\n", species\_code, order\_id);

if (order\_status == 1)

printf("Order (ID %010u) has been deleted (there were no more plants belonging to it)\n", order\_id);

} else if (ret == 0) {

printf("Nothing has changed (species %u not in order [ID %010u])\n", species\_code, order\_id);

} else {

printf("Operation failed\n");

}

printf("Press enter key to get back to menu ...\n");

getchar();

}

static int attempt\_finalize\_order(unsigned int order\_id)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[2];

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_cliente VARCHAR(16)

param[0].buffer = curr\_customer.code;

param[0].buffer\_length = strlen(curr\_customer.code);

param[1].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_ordine INT

param[1].buffer = &order\_id;

param[1].buffer\_length = sizeof(order\_id);

if (!exec\_sp(&stmt, param, "call finalizza\_ordine(?, ?)"))

return false;

mysql\_stmt\_close(stmt);

return true;

}

static void finalize\_order(void)

{

char buffer\_for\_integer[BUFFSIZE\_XS];

unsigned int order\_id;

memset(buffer\_for\_integer, 0, sizeof(buffer\_for\_integer));

init\_screen(false);

printf("\*\*\* Finalize an order \*\*\*\n");

printf("Customer code.......................................: %s\n", curr\_customer.code);

if (ask\_for\_tips("Do you wanna see a report of your open orders", 0)) {

if (!attempt\_report\_orders\_short(true))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert order id.....................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

order\_id = strtol(buffer\_for\_integer, NULL, 10);

putchar('\n');

if (attempt\_finalize\_order(order\_id))

printf("Order %010u has been finalized\n", order\_id);

else

printf("Operation failed\n");

printf("Press enter key to get back to menu ...\n");

getchar();

}

static bool attempt\_update\_addr(char \*addr, bool is\_res)

{

char sp\_str[BUFFSIZE\_L];

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[2];

memset(sp\_str, 0, sizeof(sp\_str));

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_cliente VARCHAR(16)

param[0].buffer = curr\_customer.code;

param[0].buffer\_length = strlen(curr\_customer.code);

if (strlen(addr) != 0) {

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_indirizzo VARCHAR(64)

param[1].buffer = addr;

param[1].buffer\_length = strlen(addr);

} else {

param[1].buffer\_type = MYSQL\_TYPE\_NULL; // IN var\_indirizzo VARCHAR(64)

param[1].buffer = NULL;

param[1].buffer\_length = 0;

}

snprintf(sp\_str, BUFFSIZE\_L, "call modifica\_%s(?, ?)", (is\_res) ? "residenza" : "fatturazione");

if (!exec\_sp(&stmt, param, sp\_str))

return false;

mysql\_stmt\_close(stmt);

return true;

}

static void update\_addr(bool is\_res)

{

char addr[BUFFSIZE\_M];

memset(addr, 0, sizeof(addr));

init\_screen(false);

printf("\*\*\* Update your %s address \*\*\*\n", (is\_res) ? "residential" : "billing");

printf("Customer code......%s: %s\n", (is\_res) ? "" : ".............", curr\_customer.code);

printf("Insert new address %s: ", (is\_res) ? "" : " (default null)");

get\_input(BUFFSIZE\_M, addr, false, is\_res);

putchar('\n');

if (attempt\_update\_addr(addr, is\_res))

printf("Address succesfully updated\n");

else

printf("Operation failed\n");

printf("Press enter key to get back to menu ...\n");

getchar();

}

static bool attempt\_show\_contact\_list(bool is\_customer)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[1];

char sp\_str[BUFFSIZE\_L];

memset(param, 0, sizeof(param));

memset(sp\_str, 0, sizeof(sp\_str));

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_cliente VARCHAR(16)

param[0].buffer = curr\_customer.code;

param[0].buffer\_length = strlen(curr\_customer.code);

snprintf(sp\_str, BUFFSIZE\_L, "call visualizza\_contatti\_%s(?)", (is\_customer) ? "cliente" : "referente");

if (!exec\_sp(&stmt, param, sp\_str))

return false;

if (!dump\_result\_set(stmt, "\nContact list:", LEADING\_ZERO\_BITMASK\_IDX\_0)) {

CLOSE\_AND\_RETURN(false, stmt);

}

mysql\_stmt\_close(stmt);

return true;

}

static bool attempt\_to\_modify\_contact\_list(char \*contact, bool is\_customer, bool to\_delete)

{

char sp\_str[BUFFSIZE\_L];

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[2];

memset(sp\_str, 0, sizeof(sp\_str));

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_cliente VARCHAR(16)

param[0].buffer = curr\_customer.code;

param[0].buffer\_length = strlen(curr\_customer.code);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_contatto VARCHAR(256)

param[1].buffer = contact;

param[1].buffer\_length = strlen(contact);

if (to\_delete) {

if (is\_customer)

snprintf(sp\_str, BUFFSIZE\_L, "call rimuovi\_contatto\_cliente(?, ?)");

else

snprintf(sp\_str, BUFFSIZE\_L, "call rimuovi\_contatto\_referente(?, ?)");

} else {

if (is\_customer)

snprintf(sp\_str, BUFFSIZE\_L, "call modifica\_contatto\_preferito\_cliente(?, ?)");

else

snprintf(sp\_str, BUFFSIZE\_L, "call modifica\_contatto\_preferito\_referente(?, ?)");

}

if (!exec\_sp(&stmt, param, sp\_str))

return false;

mysql\_stmt\_close(stmt);

return true;

}

static void modify\_contact\_list(bool is\_customer, bool to\_delete)

{

char contact[BUFFSIZE\_XL];

char message[BUFFSIZE\_L];

memset(contact, 0, sizeof(contact));

memset(message, 0, sizeof(message));

init\_screen(false);

if (to\_delete)

printf("\*\*\* Remove a contact from your %slist \*\*\*\n", (is\_customer) ? "" : "referent");

else

printf("\*\*\* Change %sfavourite contact %s\*\*\*\n", (is\_customer) ? "your" : "", (is\_customer) ? "" : "of your referent ");

printf("%s code....................................%s: %s\n",

(is\_customer) ? "Customer" : "Referent",

(is\_customer) ? "": ".........",

curr\_customer.code);

snprintf(message, BUFFSIZE\_L, "Do you wanna see a report of your %scontacts",

(is\_customer) ? "" : "referent ");

if (ask\_for\_tips(message, 0)) {

if (!attempt\_show\_contact\_list(is\_customer))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert contact...................................%s: ", (is\_customer) ? "": ".........");

get\_input(BUFFSIZE\_XL, contact, false, true);

putchar('\n');

if (attempt\_to\_modify\_contact\_list(contact, is\_customer, to\_delete))

printf("Contact succesfully %s\n", (to\_delete) ? "removed" : "changed");

else

printf("Operation failed\n");

printf("Press enter key to get back to menu ...\n");

getchar();

}

static bool attempt\_add\_contact(char \*contact, char \*type, bool is\_customer)

{

char sp\_str[BUFFSIZE\_L];

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[3];

memset(sp\_str, 0, sizeof(sp\_str));

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_cliente VARCHAR(16)

param[0].buffer = curr\_customer.code;

param[0].buffer\_length = strlen(curr\_customer.code);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_contatto VARCHAR(256)

param[1].buffer = contact;

param[1].buffer\_length = strlen(contact);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_tipo CHAR(16)

param[2].buffer = type;

param[2].buffer\_length = strlen(type);

snprintf(sp\_str, BUFFSIZE\_L, "call aggiungi\_contatto\_%s(?, ?, ?)",

(is\_customer) ? "cliente" : "referente");

if (!exec\_sp(&stmt, param, sp\_str))

return false;

if (!dump\_result\_set(stmt, "\nUpdated list:", 0)) {

CLOSE\_AND\_RETURN(false, stmt);

}

mysql\_stmt\_close(stmt);

return true;

}

static void add\_contact(bool is\_customer, bool show\_prompt)

{

char contact[BUFFSIZE\_XL];

char type[BUFFSIZE\_XS];

char message[BUFFSIZE\_L];

char choice;

memset(contact, 0, sizeof(contact));

memset(type, 0, sizeof(type));

memset(message, 0, sizeof(message));

init\_screen(false);

printf("\*\*\* Add a contact to your %slist \*\*\*\n", (is\_customer) ? "" : "referent ");

printf("%s code....................................%s: %s\n",

(is\_customer) ? "Customer" : "Referent",

(is\_customer) ? "": ".........",

curr\_customer.code);

snprintf(message, BUFFSIZE\_L, "Do you wanna see a report of your %scontacts", (is\_customer) ? "" : "referent ");

if (ask\_for\_tips(message, 0)) {

if (!attempt\_show\_contact\_list(is\_customer))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert new contact...............................%s: ", (is\_customer) ? "": ".........");

get\_input(BUFFSIZE\_XL, contact, false, true);

snprintf(message, BUFFSIZE\_L, "Select type [m]obile, [l]andline, [e]mail%s",

(is\_customer) ? "": ".........");

choice = multi\_choice(message, "mle", 3);

switch (choice) {

case 'm': snprintf(type, BUFFSIZE\_XS, "cellulare"); break;

case 'l': snprintf(type, BUFFSIZE\_XS, "telefono"); break;

case 'e': snprintf(type, BUFFSIZE\_XS, "email"); break;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

putchar('\n');

if (attempt\_add\_contact(contact, type, is\_customer)) {

choice = multi\_choice("\nDo you wanna set this as favourite contact?", "yn", 2);

if (choice == 'y') {

if (attempt\_to\_modify\_contact\_list(contact, is\_customer, false))

printf("Contact succesfully changed\n");

else

printf("Operation failed\n");

} else {

return;

}

} else {

printf("Operation failed\n");

}

if (show\_prompt) {

printf("Press enter key to get back to menu ...\n");

getchar();

}

}

static bool attempt\_report\_order(unsigned int order\_id)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[2];

struct order\_info order;

int i = 0;

int status;

int flags[] = {

LEADING\_ZERO\_BITMASK\_IDX\_0,

LEADING\_ZERO\_BITMASK\_IDX\_0,

0

};

char \*messages[] = {

"\nOrder info:",

"\n\nInvolved species:",

"\n\nEconiomic details:"

};

memset(&order, 0, sizeof(order));

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_cliente VARCHAR(16)

param[0].buffer = curr\_customer.code;

param[0].buffer\_length = strlen(curr\_customer.code);

param[1].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_ordine INT

param[1].buffer = &order\_id;

param[1].buffer\_length = sizeof(order\_id);

if(!exec\_sp(&stmt, param, "call report\_ordine(?, ?)"))

return false;

do {

if(conn->server\_status & SERVER\_PS\_OUT\_PARAMS)

goto next;

if (!dump\_result\_set(stmt, messages[i], flags[i])) {

CLOSE\_AND\_RETURN(false, stmt);

}

++i;

next:

status = mysql\_stmt\_next\_result(stmt);

if (status > 0) {

print\_stmt\_error(stmt, "Unexpected condition");

CLOSE\_AND\_RETURN(false, stmt);

}

} while (status == 0);

mysql\_stmt\_close(stmt);

return true;

}

static void report\_order(void)

{

char buffer\_for\_integer[BUFFSIZE\_XS];

unsigned int order\_id;

memset(buffer\_for\_integer, 0, sizeof(buffer\_for\_integer));

init\_screen(false);

printf("\*\*\* View order details \*\*\*\n");

printf("Customer code..................................: %s\n", curr\_customer.code);

if (ask\_for\_tips("Do you wanna see a report of your orders", 0)) {

if (!attempt\_report\_orders\_short(false))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert order id................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

order\_id = strtol(buffer\_for\_integer, NULL, 10);

putchar('\n');

if (!attempt\_report\_order(order\_id))

printf("Operation failed\n");

printf("\nPress enter key to get back to menu ...\n");

getchar();

}

static void order\_management\_menu(void)

{

char choice;

while (true) {

init\_screen(false);

printf("\*\*\* [ORDER MANAGEMENT] What do you wanna do? \*\*\*\n\n");

printf("1) Open a new order\n");

printf("2) Add a species to already opened order\n");

printf("3) Remove a species from an order not closed yet\n");

printf("4) Change the number of plants belonging to a species in an order\n");

printf("5) Finalize an order\n");

printf("6) Search a species by common name\n");

printf("7) View order details\n");

printf("q) Back to main menu\n");

choice = multi\_choice("Pick an option", "1234567q", 8);

switch (choice) {

case '1': open\_order(); break;

case '2': exec\_op\_on\_order(true); break;

case '3': remove\_spec\_from\_order(); break;

case '4': exec\_op\_on\_order(false); break;

case '5': finalize\_order(); break;

case '6': search\_species(); break;

case '7': report\_order(); break;

case 'q': return;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

}

}

static void profile\_management\_menu(void)

{

char choice;

while (true) {

init\_screen(false);

printf("\*\*\* [PROFILE MANAGEMENT] What do you wanna do? \*\*\*\n\n");

printf("1) Update your residential address\n");

printf("2) Update your billing address\n");

printf("3) Add a contact to your list\n");

printf("4) Remove contact from your list\n");

printf("5) Change your favourite contact\n");

printf("q) Back to main menu\n");

choice = multi\_choice("Pick an option", "12345q", 6);

switch (choice) {

case '1': update\_addr(true); break;

case '2': update\_addr(false); break;

case '3': add\_contact(true, true); break;

case '4': modify\_contact\_list(true, true); break;

case '5': modify\_contact\_list(true, false); break;

case 'q': return;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

}

}

static void referent\_management\_menu(void)

{

char choice;

while (true) {

init\_screen(false);

printf("\*\*\* [REFERENT MANAGEMENT] What do you wanna do? \*\*\*\n\n");

printf("1) Add a contact to your referent list\n");

printf("2) Remove contact from your referent list\n");

printf("3) Change favourite contact of your referent\n");

printf("q) Back to main menu\n");

choice = multi\_choice("Pick an option", "1234", 4);

switch (choice){

case '1': add\_contact(false, true); break;

case '2': modify\_contact\_list(false, true); break;

case '3': modify\_contact\_list(false, false); break;

case 'q': return;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

}

}

static void main\_menu(void)

{

char choice;

if (curr\_customer.is\_private) {

while (true) {

init\_screen(true);

printf("Welcome %s (%s)\n\n", curr\_customer.username, curr\_customer.code);

printf("\*\*\* What do you wanna do? \*\*\*\n\n");

printf("1) Orders management\n");

printf("2) Profile management\n");

printf("p) Change password\n");

printf("q) Quit\n");

choice = multi\_choice("Pick an option", "12pq", 4);

switch (choice) {

case '1': order\_management\_menu(); break;

case '2': profile\_management\_menu(); break;

case 'p': change\_password(curr\_customer.username); break;

case 'q': printf("Bye bye!\n\n\n"); return;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

}

} else {

while (true) {

init\_screen(true);

printf("Welcome %s (%s)\n\n", curr\_customer.username, curr\_customer.code);

printf("\*\*\* What do you wanna do? \*\*\*\n\n");

printf("1) Orders management\n");

printf("2) Profile management\n");

printf("3) Referent management\n");

printf("4) Change password\n");

printf("q) Quit\n");

choice = multi\_choice("Pick an option", "1234q", 5);

switch (choice) {

case '1': order\_management\_menu(); break;

case '2': profile\_management\_menu(); break;

case '3': referent\_management\_menu(); break;

case '4': change\_password(curr\_customer.username); break;

case 'q': printf("Bye bye!\n\n\n"); return;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

}

}

}

void run\_as\_customer(char \*username, char \*customer\_code, bool is\_private, bool first\_access)

{

char path[BUFFSIZE\_M];

struct configuration cnf;

char choice;

memset(path, 0, sizeof(path));

memset(&curr\_customer, 0, sizeof(curr\_customer));

memset(&cnf, 0, sizeof(cnf));

memcpy(curr\_customer.code, customer\_code, BUFFSIZE\_XS);

memcpy(curr\_customer.username, username, BUFFSIZE\_L);

curr\_customer.is\_private = is\_private;

snprintf(path, BUFFSIZE\_M, "config/%s.user", (is\_private) ? "pcs" : "rcs");

if (parse\_config(path, &cnf, "=")) {

fprintf(stderr, "Invalid configuration file selected (%s)\n", (is\_private) ? "PCS" : "RCS");

exit(EXIT\_FAILURE);

}

if(mysql\_change\_user(conn, cnf.username, cnf.password, cnf.database)) {

fprintf(stderr, "Unable to switch privileges\n");

exit(EXIT\_FAILURE);

}

if (first\_access) {

choice = multi\_choice("Do you wanna insert a contact......?", "yn", 2);

if (choice == 'y') {

add\_contact(true, false);

sleep(1.5);

}

}

main\_menu();

}

FILE: defines.h

#pragma once

#include <stdio.h>

#include <mysql.h>

#include <stdbool.h>

/\* dimensione effettiva +1 (per '\0') \*/

#define BUFFSIZE\_XS 17

#define BUFFSIZE\_S 33

#define BUFFSIZE\_M 65

#define BUFFSIZE\_L 129

#define BUFFSIZE\_XL 257

/\* bitmask \*/

#define LEADING\_ZERO\_BITMASK\_IDX\_0 1

#define LEADING\_ZERO\_BITMASK\_IDX\_1 2

#define LEADING\_ZERO\_BITMASK\_IDX\_2 4

#define CLOSE\_AND\_RETURN(retval, obj) \

mysql\_stmt\_close(obj); \

return (retval);

extern MYSQL \*conn;

struct configuration {

char host[BUFFSIZE\_L];

char username[BUFFSIZE\_L];

char password[BUFFSIZE\_L];

unsigned int port;

char database[BUFFSIZE\_L];

};

bool parse\_config(const char \*path, struct configuration \*conf, const char \*delimiter);

void print\_error(MYSQL \*conn, char \*message);

size\_t get\_input(unsigned int length, char \*string, bool hide, bool not\_null);

void init\_screen(bool);

bool setup\_prepared\_stmt(MYSQL\_STMT \*\*stmt, char \*statement, MYSQL \*conn);

void print\_stmt\_error (MYSQL\_STMT \*stmt, char \*message);

char multi\_choice(const char \*question, const char \*choices, int no\_choices);

void run\_as\_customer(char \*username, char \*customer\_code, bool is\_private, bool first\_access);

void run\_as\_manager(char \*username);

void run\_as\_chief\_of\_staff(char \*username);

void run\_as\_warehouse\_clerk(char \*username);

void run\_as\_order\_processor(char \*username);

bool dump\_result\_set(MYSQL\_STMT \*stmt, char \*title, int leading\_zeros\_bitmask);

void change\_password(char \*username);

void species\_tips(unsigned int dots);

bool ask\_for\_tips(const char \*message, unsigned int dots);

int format\_prompt(char \*dest, size\_t length, const char \*src, unsigned int dots);

bool attempt\_search\_species(bool only\_flowery, char \*name);

bool exec\_sp(MYSQL\_STMT \*\*stmt\_ptr, MYSQL\_BIND \*param, char \*sp\_name);

bool fetch\_res\_sp(MYSQL\_STMT \*stmt, MYSQL\_BIND \*param);

FILE: inout.c

#include <unistd.h>

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#include <termios.h>

#include <signal.h>

#include <stdbool.h>

static volatile sig\_atomic\_t signo;

static void handler(int s) { signo = s; }

size\_t get\_input(unsigned int length, char \*string, bool hide, bool not\_null)

{

char c;

unsigned int i;

struct sigaction sa, savealrm, saveint, savehup, savequit, saveterm, savetstp, savettin, savettou;

struct termios term, oterm;

if (hide) {

(void) fflush(stdout);

(void) sigemptyset(&sa.sa\_mask);

sa.sa\_flags = SA\_INTERRUPT;

sa.sa\_handler = handler;

(void) sigaction(SIGALRM, &sa, &savealrm);

(void) sigaction(SIGINT, &sa, &saveint);

(void) sigaction(SIGHUP, &sa, &savehup);

(void) sigaction(SIGQUIT, &sa, &savequit);

(void) sigaction(SIGTERM, &sa, &saveterm);

(void) sigaction(SIGTSTP, &sa, &savetstp);

(void) sigaction(SIGTTIN, &sa, &savettin);

(void) sigaction(SIGTTOU, &sa, &savettou);

if (tcgetattr(fileno(stdin), &oterm) == 0) {

(void) memcpy(&term, &oterm, sizeof(struct termios));

term.c\_lflag &= ~(ECHO|ECHONL);

(void) tcsetattr(fileno(stdin), TCSAFLUSH, &term);

} else {

(void) memset(&term, 0, sizeof(struct termios));

(void) memset(&oterm, 0, sizeof(struct termios));

}

}

read\_loop:

for (i = 0; i < length; ++i) {

(void) fread(&c, sizeof(char), 1, stdin);

if (c == '\n') {

string[i] = '\0';

break;

} else {

string[i] = c;

}

}

if (not\_null && i == 0)

goto read\_loop;

if (i == length - 1)

string[i] = '\0';

if (strlen(string) >= length) {

do {

c = getchar();

} while (c != '\n');

}

if (hide) {

(void) write(fileno(stdout), "\n", 1);

(void) tcsetattr(fileno(stdin), TCSAFLUSH, &oterm);

(void) sigaction(SIGALRM, &savealrm, NULL);

(void) sigaction(SIGINT, &saveint, NULL);

(void) sigaction(SIGHUP, &savehup, NULL);

(void) sigaction(SIGQUIT, &savequit, NULL);

(void) sigaction(SIGTERM, &saveterm, NULL);

(void) sigaction(SIGTSTP, &savetstp, NULL);

(void) sigaction(SIGTTIN, &savettin, NULL);

(void) sigaction(SIGTTOU, &savettou, NULL);

if (signo)

(void) raise(signo);

}

return i;

}

char multi\_choice(const char \*question, const char \*choices, int no\_choices)

{

char choices\_str[2 \* no\_choices \* sizeof(char)];

int i, j = 0;

for (i = 0; i < no\_choices; ++i) {

choices\_str[j++] = choices[i];

choices\_str[j++] = '/';

}

choices\_str[j-1] = '\0';

while (true) {

printf("%s [%s]: ", question, choices\_str);

char c;

get\_input(1, &c, false, true);

c = tolower(c);

for (i = 0; i < no\_choices; ++i) {

if(c == tolower(choices[i]))

return c;

}

printf("Sorry not compliant input, please retry!\n");

}

}

FILE: main.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

#include <mysql.h>

#include "defines.h"

enum role {

PCS, // Customer (private)

RCS, // Customer (retailer)

WHC, // Warehouse clerk

OPC, // Order processor

MNG, // Manager

COS, // Chief of staff

ERR // ERROR

};

struct credentials {

char username[BUFFSIZE\_L];

char password[BUFFSIZE\_L];

};

struct customer\_signup\_params {

struct credentials \*creds;

char code[BUFFSIZE\_XS];

char name[BUFFSIZE\_S];

char residential\_address[BUFFSIZE\_M];

char billing\_address[BUFFSIZE\_M];

char referent\_first\_name[BUFFSIZE\_S];

char referent\_last\_name[BUFFSIZE\_S];

};

MYSQL \*conn;

static enum role attempt\_login(struct credentials \*cred, char \*identifier)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[4];

int role;

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_username VARCHAR(128)

param[0].buffer = cred->username;

param[0].buffer\_length = strlen(cred->username);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_password VARCHAR(128)

param[1].buffer = cred->password;

param[1].buffer\_length = strlen(cred->password);

param[2].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_ruolo INT

param[2].buffer = &role;

param[2].buffer\_length = sizeof(role);

param[3].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // OUT var\_codice\_cliente VARCHAR(16)

param[3].buffer = identifier;

param[3].buffer\_length = BUFFSIZE\_XS \* sizeof(char);

if (!exec\_sp(&stmt, param, "call login(?, ?, ?, ?)"))

return ERR;

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_ruolo INT

param[0].buffer = &role;

param[0].buffer\_length = sizeof(role);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // OUT var\_codice\_cliente VARCHAR(16)

param[1].buffer = identifier;

param[1].buffer\_length = BUFFSIZE\_XS \* sizeof(char);

if (!fetch\_res\_sp(stmt, param))

return ERR;

mysql\_stmt\_close(stmt);

return role;

}

static bool attempt\_signup(struct customer\_signup\_params \*cst, bool is\_private)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[(is\_private ? 6 : 8)];

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_codice CHAR(16) / CHAR(11)

param[0].buffer = cst->code;

param[0].buffer\_length = strlen(cst->code);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_nome VARCHAR(32)

param[1].buffer = cst->name;

param[1].buffer\_length = strlen(cst->name);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_res VARCHAR(64)

param[2].buffer = cst->residential\_address;

param[2].buffer\_length = strlen(cst->residential\_address);

if (strlen(cst->billing\_address) > 0) {

param[3].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_fat VARCHAR(64)

param[3].buffer = cst->billing\_address;

param[3].buffer\_length = strlen(cst->billing\_address);

} else {

param[3].buffer\_type = MYSQL\_TYPE\_NULL; // IN var\_fat VARCHAR(64)

param[3].buffer = NULL;

param[3].buffer\_length = 0;

}

param[4].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_user VARCHAR(128)

param[4].buffer = (cst->creds)->username;

param[4].buffer\_length = strlen((cst->creds)->username);

param[5].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_pass VARCHAR(128)

param[5].buffer = (cst->creds)->password;

param[5].buffer\_length = strlen((cst->creds)->password);

if (!is\_private) {

param[6].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_nome\_ref VARCHAR(32)

param[6].buffer = cst->referent\_first\_name;

param[6].buffer\_length = strlen(cst->referent\_first\_name);

param[7].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_cognome\_ref VARCHAR(32)

param[7].buffer = cst->referent\_last\_name;

param[7].buffer\_length = strlen(cst->referent\_last\_name);

}

if (is\_private) {

if (!exec\_sp(&stmt, param, "call registra\_privato(?, ?, ?, ?, ?, ?)"))

return false;

} else {

if (!exec\_sp(&stmt, param, "call registra\_rivendita(?, ?, ?, ?, ?, ?, ?, ?)"))

return false;

}

mysql\_stmt\_close(stmt);

return true;

}

static bool attempt\_change\_password(char \*username, char \*old\_passwd, char \*new\_passwd)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[3];

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_username VARCHAR(128)

param[0].buffer = username;

param[0].buffer\_length = strlen(username);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_vecchia\_password VARCHAR(128)

param[1].buffer = old\_passwd;

param[1].buffer\_length = strlen(old\_passwd);

param[2].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_nuova\_password VARCHAR(128)

param[2].buffer = new\_passwd;

param[2].buffer\_length = strlen(new\_passwd);

if (!exec\_sp(&stmt, param, "call modifica\_password(?, ?, ?)"))

return false;

mysql\_stmt\_close(stmt);

return true;

}

static bool login\_manager(void)

{

char client\_identifier[BUFFSIZE\_XS];

struct credentials cred;

enum role role;

char choice;

memset(&client\_identifier, 0, sizeof(client\_identifier));

memset(&cred, 0, sizeof(cred));

init\_screen(false);

printf("Insert username: ");

get\_input(BUFFSIZE\_L, cred.username, false, true);

printf("Insert password: ");

get\_input(BUFFSIZE\_L, cred.password, true, true);

role = attempt\_login(&cred, client\_identifier);

switch (role) {

case PCS: run\_as\_customer(cred.username, client\_identifier, true, false); break;

case RCS: run\_as\_customer(cred.username, client\_identifier, false, false); break;

case MNG: run\_as\_manager(cred.username); break;

case COS: run\_as\_chief\_of\_staff(cred.username); break;

case WHC: run\_as\_warehouse\_clerk(cred.username); break;

case OPC: run\_as\_order\_processor(cred.username); break;

case ERR: printf("Login failed!\n"); break;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

if (role == ERR) {

choice = multi\_choice("Do you wanna quit? ", "yn", 2);

return (choice == 'y');

}

return true;

}

static char \*strupper(char \*str)

{

for (unsigned int i = 0; i < strlen(str); ++i)

str[i] = toupper(str[i]);

return str;

}

static bool signup\_manager(void)

{

char password\_check[BUFFSIZE\_L];

struct customer\_signup\_params cst;

struct credentials creds;

char modality;

char choice;

memset(&cst, 0, sizeof(cst));

memset(&creds, 0, sizeof(creds));

memset(password\_check, 0, sizeof(password\_check));

init\_screen(false);

modality = multi\_choice("Are you a [p]rivate or [r]etailer..?", "pr", 2);

printf("Insert username...........................: ");

get\_input(BUFFSIZE\_L, creds.username, false, true);

retype\_pass:

printf("Insert password...........................: ");

get\_input(BUFFSIZE\_L, creds.password, true, true);

printf("Confirm password..........................: ");

get\_input(BUFFSIZE\_L, password\_check, true, true);

if (strcmp(creds.password, password\_check) != 0) {

printf("Mismatch password, please retry!\n");

goto retype\_pass;

}

if (modality == 'p') {

printf("Insert fiscal code........................: ");

get\_input(16, cst.code, false, true);

} else {

printf("Insert VAT code...........................: ");

get\_input(11, cst.code, false, true);

}

printf("Insert your name..........................: ");

get\_input(BUFFSIZE\_S, cst.name, false, true);

printf("Insert your residential address...........: ");

get\_input(BUFFSIZE\_M, cst.residential\_address, false, true);

printf("Insert your billing address (default null): ");

get\_input(BUFFSIZE\_M, cst.billing\_address, false, false);

if (modality == 'r') {

printf("Insert referent first name................: ");

get\_input(BUFFSIZE\_S, cst.referent\_first\_name, false, true);

printf("Insert referent last name:................:");

get\_input(BUFFSIZE\_S, cst.referent\_last\_name, false, true);

}

cst.creds = &creds;

if (!attempt\_signup(&cst, (modality == 'p'))) {

printf("Signup failed!\n");

choice = multi\_choice("Do you wanna quit?", "yn", 2);

return (choice == 'y');

}

run\_as\_customer(creds.username, strupper(cst.code), (modality == 'p'), true);

return true;

}

void change\_password(char \*username)

{

char old\_passwd[BUFFSIZE\_L];

char new\_passwd[BUFFSIZE\_L];

char passwd\_check[BUFFSIZE\_L];

memset(old\_passwd, 0, sizeof(old\_passwd));

memset(new\_passwd, 0, sizeof(new\_passwd));

memset(passwd\_check, 0, sizeof(passwd\_check));

init\_screen(false);

printf("\*\*\* Change password \*\*\*\n");

printf("Customer username......: %s\n", username);

printf("Insert old password....: ");

get\_input(BUFFSIZE\_L, old\_passwd, true, true);

retype\_pass:

printf("Insert new password....: ");

get\_input(BUFFSIZE\_L, new\_passwd, true, true);

printf("Retype new password....: ");

get\_input(BUFFSIZE\_L, passwd\_check, true, true);

if (strcmp(new\_passwd, passwd\_check) != 0) {

printf("Mismatch password, please retry!\n");

goto retype\_pass;

}

if (attempt\_change\_password(username, old\_passwd, new\_passwd))

printf("Password has been changed!\n");

else

printf("Operation failed\n");

printf("Press enter key to get back to menu ...\n");

getchar();

}

int format\_prompt(char \*dest, size\_t length, const char \*src, unsigned int dots)

{

int len = snprintf(dest, length, src);

for (unsigned int i = 0; i < dots; ++i)

dest[len + i] = '.';

dest[len + dots] = '?';

return len + dots;

}

bool ask\_for\_tips(const char \*message, unsigned int dots)

{

char prompt[BUFFSIZE\_XL];

char choice;

memset(prompt, 0, sizeof(prompt));

putchar('\n');

format\_prompt(prompt, BUFFSIZE\_XL, message, dots);

choice = multi\_choice(prompt, "yn", 2);

return (choice == 'y');

}

bool attempt\_search\_species(bool only\_flowery, char \*name)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[2];

char prompt[BUFFSIZE\_L];

memset(param, 0, sizeof(param));

memset(prompt, 0, sizeof(prompt));

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_nome\_comune VARCHAR(64)

param[0].buffer = name;

param[0].buffer\_length = strlen(name);

param[1].buffer\_type = MYSQL\_TYPE\_TINY; // IN var\_status TINYINT

param[1].buffer = &(only\_flowery);

param[1].buffer\_length = sizeof(only\_flowery);

if (!exec\_sp(&stmt, param, "call visualizza\_dettagli\_specie(?, ?)"))

return false;

if (strlen(name) > 0)

snprintf(prompt, BUFFSIZE\_L, "\nSearch results for \'%s\':", name);

else

snprintf(prompt, BUFFSIZE\_L, "\nSearch results:");

if (!dump\_result\_set(stmt, prompt, LEADING\_ZERO\_BITMASK\_IDX\_0)) {

CLOSE\_AND\_RETURN(false, stmt);

}

mysql\_stmt\_close(stmt);

return true;

}

bool exec\_sp(MYSQL\_STMT \*\*stmt\_ptr, MYSQL\_BIND \*param, char \*sp\_name)

{

if(!setup\_prepared\_stmt(stmt\_ptr, sp\_name, conn)) {

print\_stmt\_error(\*stmt\_ptr, "Unable to initialize the statement\n");

return false;

}

if (mysql\_stmt\_bind\_param(\*stmt\_ptr, param) != 0) {

print\_stmt\_error(\*stmt\_ptr, "Could not bind parameters for the statement");

CLOSE\_AND\_RETURN(false, \*stmt\_ptr);

}

if (mysql\_stmt\_execute(\*stmt\_ptr) != 0) {

print\_stmt\_error(\*stmt\_ptr, "Could not execute the statement");

CLOSE\_AND\_RETURN(false, \*stmt\_ptr);

}

return true;

}

bool fetch\_res\_sp(MYSQL\_STMT \*stmt, MYSQL\_BIND \*param)

{

if(mysql\_stmt\_bind\_result(stmt, param)) {

print\_stmt\_error(stmt, "Could not retrieve output parameter");

CLOSE\_AND\_RETURN(false, stmt);

}

if(mysql\_stmt\_fetch(stmt)) {

print\_stmt\_error(stmt, "Could not buffer results");

CLOSE\_AND\_RETURN(false, stmt);

}

return true;

}

int main(void)

{

char choice;

struct configuration cnf;

MYSQL \*ret;

memset(&cnf, 0, sizeof(cnf));

if (parse\_config("config/nrg.user", &cnf, "=")) {

fprintf(stderr, "Invalid configuration file selected (NRG)\n");

exit(EXIT\_FAILURE);

}

conn = mysql\_init(NULL);

if (conn == NULL) {

fprintf(stderr, "Out of memory, connection was not established.");

exit(EXIT\_FAILURE);

}

ret = mysql\_real\_connect(conn, cnf.host, cnf.username, cnf.password, cnf.database,

cnf.port, NULL, CLIENT\_MULTI\_STATEMENTS | CLIENT\_MULTI\_RESULTS);

if (ret == NULL) {

print\_error(conn, "Something went wrong, connection was not established.");

mysql\_close (conn);

exit(EXIT\_FAILURE);

}

while (true) {

init\_screen(false);

choice = multi\_choice("Do you wanna [l]ogin or [s]ignup?", "ls", 2);

if (choice == 'l') {

if (login\_manager())

break;

}

else if (choice == 's') {

if (signup\_manager())

break;

}

else {

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

}

mysql\_close(conn);

mysql\_library\_end();

exit(EXIT\_SUCCESS);

}

FILE manager.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

#include <mysql.h>

#include <regex.h>

#include <time.h>

#include "defines.h"

struct insert\_species\_sp\_params {

char common\_name[BUFFSIZE\_M];

char latin\_name[BUFFSIZE\_M];

signed char in\_or\_out;

char coloring[BUFFSIZE\_S];

signed char exotic;

unsigned int stock;

char price[BUFFSIZE\_XS];

};

static char curr\_user[BUFFSIZE\_L];

static bool attempt\_show\_colors(unsigned int species\_code)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[1];

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_ordine INT

param[0].buffer = &species\_code;

param[0].buffer\_length = sizeof(species\_code);

if(!exec\_sp(&stmt, param, "call visualizza\_colorazioni(?)"))

return false;

if (!dump\_result\_set(stmt, "\nColors available for selected species:", 0)) {

CLOSE\_AND\_RETURN(false, stmt);

}

mysql\_stmt\_close(stmt);

return true;

}

static int check\_price(char \*inserted\_price, char \*strerror, size\_t strerror\_length)

{

regex\_t reg;

char err\_mess[BUFFSIZE\_L];

int ret;

ret = regcomp(&reg, "^([0-9]{0,5}((.)[0-9]{0,2}))$", REG\_EXTENDED);

if (ret) {

regerror(ret, &reg, err\_mess, sizeof(err\_mess));

snprintf(strerror, strerror\_length, "Unable to compile regexp to check inserted price\nThe error was: %s\n", err\_mess);

return -1;

}

ret = regexec(&reg, inserted\_price, 0, NULL, 0);

if (ret == 0) {

regfree(&reg);

return 1;

} else if (ret == REG\_NOMATCH) {

regfree(&reg);

return 0;

} else {

regerror(ret, &reg, err\_mess, sizeof(err\_mess));

snprintf(strerror, strerror\_length, "Unable to exec regexp to check inserted price\nThe error was: %s\n", err\_mess);

return -1;

}

}

static unsigned int attempt\_insert\_species(struct insert\_species\_sp\_params \*input)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[8];

unsigned int code;

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_nome\_comune VARCHAR(64)

param[0].buffer = input->common\_name;

param[0].buffer\_length = strlen(input->common\_name);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_nome\_latino VARCHAR(64)

param[1].buffer = input->latin\_name;

param[1].buffer\_length = strlen(input->latin\_name);

param[2].buffer\_type = MYSQL\_TYPE\_TINY; // IN var\_int\_est TINYINT

param[2].buffer = &(input->in\_or\_out);

param[2].buffer\_length = sizeof(input->in\_or\_out);

if (strlen(input->coloring) > 0) {

param[3].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_colorazione VARCHAR(32)

param[3].buffer = input->coloring;

param[3].buffer\_length = strlen(input->coloring);

} else {

param[3].buffer\_type = MYSQL\_TYPE\_NULL; // IN var\_colorazione VARCHAR(32)

param[3].buffer = NULL;

param[3].buffer\_length = 0;

}

param[4].buffer\_type = MYSQL\_TYPE\_TINY; // IN var\_esotica TINYINT

param[4].buffer = &(input->exotic);

param[4].buffer\_length = sizeof(input->exotic);

param[5].buffer\_type= MYSQL\_TYPE\_LONG; // IN var\_giacenza INT

param[5].buffer = &(input->stock);

param[5].buffer\_length = sizeof(input->stock);

param[6].buffer\_type = MYSQL\_TYPE\_NEWDECIMAL; // IN var\_prezzo DECIMAL(7, 2)

param[6].buffer = input->price;

param[6].buffer\_length = strlen(input->price);

param[7].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_codice INT

param[7].buffer = &code;

param[7].buffer\_length = sizeof(code);

if (!exec\_sp(&stmt, param, "call inserisci\_nuova\_specie(?, ?, ?, ?, ?, ?, ?, ?)"))

return 0;

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_codice INT

param[0].buffer = &code;

param[0].buffer\_length = sizeof(code);

if (!fetch\_res\_sp(stmt, param))

return 0;

mysql\_stmt\_close(stmt);

return code;

}

static void insert\_a\_species(void)

{

struct insert\_species\_sp\_params params;

char buffer\_for\_integer[BUFFSIZE\_XS];

char strerror[BUFFSIZE\_XL];

unsigned int species\_code;

char choice;

int ret;

memset(&params, 0, sizeof(params));

memset(buffer\_for\_integer, 0, sizeof(buffer\_for\_integer));

memset(strerror, 0, sizeof(strerror));

init\_screen(false);

printf("\*\*\* Insert a new species \*\*\*\n");

printf("Insert common name......................................: ");

get\_input(BUFFSIZE\_M, params.common\_name, false, true);

printf("Insert latin name.......................................: ");

get\_input(BUFFSIZE\_M, params.latin\_name, false, true);

choice = multi\_choice("Which kind of species is it? [o]utdoor or [i]ndoor", "oi", 2);

params.in\_or\_out = (choice == 'i');

choice = multi\_choice("Which kind of species is it? [g]reen or [f]lowery.", "gf", 2);

if (choice == 'f') {

printf("Insert default coloring.................................: ");

get\_input(BUFFSIZE\_S, params.coloring, false, true);

}

choice = multi\_choice("Is it exotic.....................................?", "yn", 2);

params.exotic = (choice == 'y') ? 1 : 0;

printf("Insert initial stock....................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

params.stock = strtol(buffer\_for\_integer, NULL, 10);

insert\_price:

printf("Insert price (#####.##).................................: ");

get\_input(BUFFSIZE\_XS, params.price, false, true);

ret = check\_price(params.price, strerror, BUFFSIZE\_XL);

if (ret == 0) {

printf("Not compliant input please retry\n");

goto insert\_price;

}

if (ret == -1) {

fprintf(stderr, strerror);

goto exit;

}

putchar('\n');

species\_code = attempt\_insert\_species(&params);

if (species\_code > 0)

printf("New species inserted (CODE: %010u)\n", species\_code);

else

printf("Operation failed\n");

exit:

printf("Press enter key to get back to menu ...\n");

getchar();

}

static int attempt\_remove\_species(unsigned int species\_code)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[2];

int affected\_rows;

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_codice INT

param[0].buffer = &species\_code;

param[0].buffer\_length = sizeof(species\_code);

param[1].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_eliminazione\_effettiva INT

param[1].buffer = &affected\_rows;

param[1].buffer\_length = sizeof(affected\_rows);

if (!exec\_sp(&stmt, param, "call rimuovi\_specie\_di\_pianta(?, ?)"))

return -1;

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_eliminazione\_effettiva INT

param[0].buffer = &affected\_rows;

param[0].buffer\_length = sizeof(affected\_rows);

if (!fetch\_res\_sp(stmt, param))

return -1;

mysql\_stmt\_close(stmt);

return affected\_rows;

}

static void remove\_a\_species(void)

{

char buffer\_for\_integer[BUFFSIZE\_XS];

char spec\_name[BUFFSIZE\_M];

unsigned int species\_code;

int ret;

memset(buffer\_for\_integer, 0, sizeof(buffer\_for\_integer));

memset(spec\_name, 0, sizeof(spec\_name));

init\_screen(false);

printf("\*\*\* Remove a species \*\*\*\n");

if (ask\_for\_tips("Do you wanna search species by name to find the right code", 0)) {

printf("\nInsert the name to filter on (default all).......................: ");

get\_input(BUFFSIZE\_M, spec\_name, false, false);

if (!attempt\_search\_species(false, spec\_name))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert species code..............................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

species\_code = strtol(buffer\_for\_integer, NULL, 10);

putchar('\n');

ret = attempt\_remove\_species(species\_code);

if (ret > 0)

printf("Species %u succesfully deleted\n", species\_code);

else if (ret == 0)

printf("Nothing has changed (species %u not found)\n", species\_code);

else

printf("Operation failed\n");

printf("Press enter key to get back to menu ...\n");

getchar();

}

static int attempt\_add\_coloring(unsigned int species\_code, char \*coloring)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[2];

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_specie\_fiorita INT

param[0].buffer = &species\_code;

param[0].buffer\_length = sizeof(species\_code);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_colore VARCHAR(32)

param[1].buffer = coloring;

param[1].buffer\_length = strlen(coloring);

if(!exec\_sp(&stmt, param, "call aggiungi\_colorazione(?, ?)"))

return false;

if (!dump\_result\_set(stmt, "\nUpdated coloring list:", 0)) {

CLOSE\_AND\_RETURN(false, stmt);

}

mysql\_stmt\_close(stmt);

return true;

}

static void add\_coloring(void)

{

unsigned int species\_code;

char coloring[BUFFSIZE\_S];

char buffer\_for\_integer[BUFFSIZE\_XS];

char spec\_name[BUFFSIZE\_M];

memset(coloring, 0, sizeof(coloring));

memset(buffer\_for\_integer, 0, sizeof(buffer\_for\_integer));

memset(spec\_name, 0, sizeof(spec\_name));

init\_screen(false);

printf("\*\*\* Add a coloring for a flowering species \*\*\*\n");

if (ask\_for\_tips("Do you wanna search flowery species by name to find the right code", 0)) {

printf("\nInsert the name to filter on (default all)...............................: ");

get\_input(BUFFSIZE\_M, spec\_name, false, false);

if (!attempt\_search\_species(true, spec\_name))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert species code......................................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

species\_code = strtol(buffer\_for\_integer, NULL, 10);

if (ask\_for\_tips("Do you wanna see a report of available colors", 21)) {

if (!attempt\_show\_colors(species\_code))

printf("Operation failed\n");

putchar('\n');;

}

printf("Insert coloring for this species.........................................: ");

get\_input(BUFFSIZE\_S, coloring, false, true);

putchar('\n');

if (attempt\_add\_coloring(species\_code, coloring))

printf("\nColoring \"%s\" for %010u succesfully added\n", coloring, species\_code);

else

printf("\nOperation failed\n");

printf("Press enter key to get back to menu ...\n");

getchar();

}

static int attempt\_remove\_coloring(unsigned int species\_code, char \*coloring)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[3];

int affected\_rows;

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_specie\_fiorita INT

param[0].buffer = &species\_code;

param[0].buffer\_length = sizeof(species\_code);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_colore VARCHAR(32)

param[1].buffer = coloring;

param[1].buffer\_length = strlen(coloring);

param[2].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_eliminazione\_effettiva INT

param[2].buffer = &affected\_rows;

param[2].buffer\_length = sizeof(affected\_rows);

if (!exec\_sp(&stmt, param, "call rimuovi\_colorazione(?, ?, ?)"))

return -1;

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_eliminazione\_effettiva INT

param[0].buffer = &affected\_rows;

param[0].buffer\_length = sizeof(affected\_rows);

if (!fetch\_res\_sp(stmt, param))

return -1;

mysql\_stmt\_close(stmt);

return affected\_rows;

}

static void remove\_coloring(void)

{

unsigned int species\_code;

char coloring[BUFFSIZE\_S];

char buffer\_for\_integer[BUFFSIZE\_XS];

char spec\_name[BUFFSIZE\_M];

int ret;

memset(coloring, 0, sizeof(coloring));

memset(buffer\_for\_integer, 0, sizeof(buffer\_for\_integer));

memset(spec\_name, 0, sizeof(spec\_name));

init\_screen(false);

printf("\*\*\* Remove a coloring from a flowering species \*\*\*\n");

if (ask\_for\_tips("Do you wanna search flowery species by name to find the right code", 0)) {

printf("\nInsert the name to filter on (default all)...............................: ");

get\_input(BUFFSIZE\_M, spec\_name, false, false);

if (!attempt\_search\_species(true, spec\_name))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert species code......................................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

species\_code = strtol(buffer\_for\_integer, NULL, 10);

if (ask\_for\_tips("Do you wanna see a report of available colors", 21)) {

if (!attempt\_show\_colors(species\_code))

printf("Operation failed\n");

putchar('\n');;

}

printf("Insert coloring to be removed............................................: ");

get\_input(BUFFSIZE\_S, coloring, false, true);

putchar('\n');

ret = attempt\_remove\_coloring(species\_code, coloring);

if (ret > 0)

printf("Coloring \"%s\" succesfully removed for %010u\n", coloring, species\_code);

else if (ret == 0)

printf("Nothing has changed (species %010u does not have this coloring)\n", species\_code);

else

printf("Operation failed\n");

printf("Press enter key to get back to menu ...\n");

getchar();

}

static int attempt\_change\_price(unsigned int species\_code, char \*price)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[3];

int affected\_rows;

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_specie INT

param[0].buffer = &species\_code;

param[0].buffer\_length = sizeof(species\_code);

param[1].buffer\_type = MYSQL\_TYPE\_NEWDECIMAL; // IN var\_prezzo DECIMAL(7, 2)

param[1].buffer = price;

param[1].buffer\_length = strlen(price);

param[2].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_aggiornamento\_effettivo INT

param[2].buffer = &affected\_rows;

param[2].buffer\_length = sizeof(affected\_rows);

if (!exec\_sp(&stmt, param, "call modifica\_prezzo(?, ?, ?)"))

return -1;

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_aggiornamento\_effettivo INT

param[0].buffer = &affected\_rows;

param[0].buffer\_length = sizeof(affected\_rows);

if (!fetch\_res\_sp(stmt, param))

return -1;

mysql\_stmt\_close(stmt);

return affected\_rows;

}

static void change\_price(void)

{

char buffer\_for\_integer[BUFFSIZE\_XS];

char spec\_name[BUFFSIZE\_M];

char price[BUFFSIZE\_XS];

char strerror[BUFFSIZE\_XL];

unsigned int species\_code;

int ret;

memset(buffer\_for\_integer, 0, sizeof(buffer\_for\_integer));

memset(strerror, 0, sizeof(strerror));

memset(price, 0, sizeof(price));

memset(spec\_name, 0, sizeof(spec\_name));

init\_screen(false);

printf("\*\*\* Change the price of a species \*\*\*\n");

if (ask\_for\_tips("Do you wanna search species by name to find the right code", 0)) {

printf("\nInsert the name to filter on (default all).......................: ");

get\_input(BUFFSIZE\_M, spec\_name, false, false);

if (!attempt\_search\_species(false, spec\_name))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert species code.............................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

species\_code = strtol(buffer\_for\_integer, NULL, 10);

insert\_price:

printf("Insert price (#####.##).........................................: ");

get\_input(BUFFSIZE\_XS, price, false, true);

ret = check\_price(price, strerror, BUFFSIZE\_XL);

if (ret == 0) {

printf("Not compliant input please retry\n");

goto insert\_price;

}

if (ret == -1) {

fprintf(stderr, strerror);

goto exit;

}

putchar('\n');

ret = attempt\_change\_price(species\_code, price);

if (ret > 0)

printf("Price updated for %010u\n", species\_code);

else if (ret == 0)

printf("Nothing has changed (the stored and entered prices coincide)\n");

else

printf("Operation failed\n");

exit:

printf("Press enter key to get back to menu ...\n");

getchar();

}

static bool attempt\_report\_species(unsigned int species\_code)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[1];

char prompt[2 \* BUFFSIZE\_XL];

int i = 0;

int status;

time\_t t = time(NULL);

struct tm \*tm = localtime(&t);

memset(prompt, 0, sizeof(prompt));

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_specie INT

param[0].buffer = &species\_code;

param[0].buffer\_length = sizeof(species\_code);

if(!exec\_sp(&stmt, param, "call report\_specie(?)"))

return false;

do {

if(conn->server\_status & SERVER\_PS\_OUT\_PARAMS)

goto next;

if (i == 0) {

if (!dump\_result\_set(stmt, "\nSpecies info:", LEADING\_ZERO\_BITMASK\_IDX\_0)) {

CLOSE\_AND\_RETURN(false, stmt);

}

} else {

snprintf(prompt, 2 \* BUFFSIZE\_XL, "\n\nSales details (year: %d):", (tm->tm\_year + 1900));

if (!dump\_result\_set(stmt, prompt, 0)) {

CLOSE\_AND\_RETURN(false, stmt);

}

}

++i;

next:

status = mysql\_stmt\_next\_result(stmt);

if (status > 0) {

print\_stmt\_error(stmt, "Unexpected condition");

CLOSE\_AND\_RETURN(false, stmt);

}

} while (status == 0);

mysql\_stmt\_close(stmt);

return true;

}

static void report\_species(void)

{

unsigned int species\_code;

char buffer\_for\_integer[BUFFSIZE\_XS];

char spec\_name[BUFFSIZE\_M];

memset(buffer\_for\_integer, 0, sizeof(buffer\_for\_integer));

memset(spec\_name, 0, sizeof(spec\_name));

init\_screen(false);

printf("\*\*\* View sales trend for a chosen species \*\*\*\n");

if (ask\_for\_tips("Do you wanna search species by name to find the right code", 0)) {

printf("\nInsert the name to filter on (default all).......................: ");

get\_input(BUFFSIZE\_M, spec\_name, false, false);

if (!attempt\_search\_species(false, spec\_name))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert species code..............................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

species\_code = strtol(buffer\_for\_integer, NULL, 10);

putchar('\n');

if (!attempt\_report\_species(species\_code))

printf("Operation failed\n");

printf("\nPress enter key to get back to menu ...\n");

getchar();

}

void run\_as\_manager(char \*username)

{

struct configuration cnf;

char choice;

memset(&cnf, 0, sizeof(cnf));

memset(curr\_user, 0, sizeof(curr\_user));

strncpy(curr\_user, username, BUFFSIZE\_L);

if (parse\_config("config/mng.user", &cnf, "=")) {

fprintf(stderr, "Invalid configuration file selected (MNG)\n");

exit(EXIT\_FAILURE);

}

if (mysql\_change\_user(conn, cnf.username, cnf.password, cnf.database)) {

fprintf(stderr, "Unable to switch privileges\n");

exit(EXIT\_FAILURE);

}

while (true) {

init\_screen(true);

printf("Welcome %s\n\n", curr\_user);

printf("\*\*\* What do you wanna do? \*\*\*\n\n");

printf("1) Insert a new species\n");

printf("2) Remove a species\n");

printf("3) Add a coloring for a flowering species\n");

printf("4) Remove a coloring from a flowering species list\n");

printf("5) Change the price of a species\n");

printf("6) View sales trend for a chosen species\n");

printf("p) Change password\n");

printf("q) Quit\n");

choice = multi\_choice("Pick an option", "123456pq", 8);

switch (choice) {

case '1': insert\_a\_species(); break;

case '2': remove\_a\_species(); break;

case '3': add\_coloring(); break;

case '4': remove\_coloring(); break;

case '5': change\_price(); break;

case '6': report\_species(); break;

case 'p': change\_password(curr\_user); break;

case 'q': printf("Bye bye!\n\n\n"); return;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

}

}

FILE orderprocessor.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

#include <mysql.h>

#include "defines.h"

struct create\_pack\_sp\_param {

unsigned int order\_id;

unsigned int pack\_number;

unsigned int species\_code;

unsigned int quantity;

};

static char curr\_user[BUFFSIZE\_L];

static bool attempt\_search\_species\_belonging\_to\_order(unsigned int order\_id)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[1];

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_ordine INT

param[0].buffer = &order\_id;

param[0].buffer\_length = sizeof(order\_id);

if(!exec\_sp(&stmt, param, "call visualizza\_piante\_rimanenti\_da\_impacchettare(?)"))

return false;

if (!dump\_result\_set(stmt, "Species belonging to selected order:", LEADING\_ZERO\_BITMASK\_IDX\_0)) {

CLOSE\_AND\_RETURN(false, stmt);

}

mysql\_stmt\_close(stmt);

return true;

}

static bool attempt\_show\_status(unsigned int order\_id)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[1];

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_ordine INT

param[0].buffer = &(order\_id);

param[0].buffer\_length = sizeof(order\_id);

if(!exec\_sp(&stmt, param, "call visualizza\_stato\_ordine(?)"))

return false;

if (!dump\_result\_set(stmt, "Order info:", LEADING\_ZERO\_BITMASK\_IDX\_0)) {

CLOSE\_AND\_RETURN(false, stmt);

}

mysql\_stmt\_close(stmt);

return true;

}

static bool attempt\_exec\_op\_on\_pack(struct create\_pack\_sp\_param \*args, bool is\_create)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[4];

char sp\_str[BUFFSIZE\_L];

memset(param, 0, sizeof(param));

memset(sp\_str, 0, sizeof(sp\_str));

snprintf(sp\_str, BUFFSIZE\_L, "call %s\_pacco(?, ?, ?, ?)", (is\_create) ? "crea" : "aggiungi\_specie\_a");

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_ordine INT

param[0].buffer = &(args->order\_id);

param[0].buffer\_length = sizeof(args->order\_id);

param[1].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_specie INT

param[1].buffer = &(args->species\_code);

param[1].buffer\_length = sizeof(args->species\_code);

param[2].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_quantita INT

param[2].buffer = &(args->quantity);

param[2].buffer\_length = sizeof(args->quantity);

param[3].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_numero INT

param[3].buffer = &(args->pack\_number);

param[3].buffer\_length = sizeof(args->pack\_number);

if(!exec\_sp(&stmt, param, sp\_str))

return false;

if (is\_create) {

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_numero INT

param[0].buffer = &(args->pack\_number);

param[0].buffer\_length = sizeof(args->pack\_number);

if (!fetch\_res\_sp(stmt, param))

return false;

}

mysql\_stmt\_close(stmt);

return true;

}

static bool attempt\_report\_packs(unsigned int order\_id)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[1];

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_ordine INT

param[0].buffer = &order\_id;

param[0].buffer\_length = sizeof(order\_id);

if(!exec\_sp(&stmt, param, "call report\_pacchi(?)"))

return false;

if (!dump\_result\_set(stmt, "Processing details", LEADING\_ZERO\_BITMASK\_IDX\_0)) {

CLOSE\_AND\_RETURN(false, stmt);

}

mysql\_stmt\_close(stmt);

return true;

}

static void report\_packs(void)

{

char buffer\_for\_integer[BUFFSIZE\_XS];

unsigned int order\_id;

memset(buffer\_for\_integer, 0, sizeof(buffer\_for\_integer));

init\_screen(false);

printf("\*\*\* View details about order processing \*\*\*\n");

printf("Insert order ID......................................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

order\_id = strtol(buffer\_for\_integer, NULL, 10);

putchar('\n');

if (!attempt\_show\_status(order\_id))

printf("Operation failed\n");

putchar('\n');

if (!attempt\_search\_species\_belonging\_to\_order(order\_id))

printf("Operation failed\n");

putchar('\n');

if (!attempt\_report\_packs(order\_id))

printf("Operation failed\n");

putchar('\n');

printf("Press enter key to get back to menu ...\n");

getchar();

}

static void create\_pack(void)

{

char buffer\_for\_integer[BUFFSIZE\_XS];

char spec\_name[BUFFSIZE\_M];

struct create\_pack\_sp\_param args;

memset(buffer\_for\_integer, 0, sizeof(buffer\_for\_integer));

memset(&args, 0, sizeof(args));

memset(spec\_name, 0, sizeof(spec\_name));

init\_screen(false);

printf("\*\*\* Create new pack \*\*\*\n");

printf("Insert order ID......................................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

args.order\_id = strtol(buffer\_for\_integer, NULL, 10);

if (!attempt\_show\_status(args.order\_id))

printf("Operation failed\n");

if (ask\_for\_tips("Do you wanna see a list of species belonging to selected order", 0)) {

if (!attempt\_search\_species\_belonging\_to\_order(args.order\_id))

printf("Operation failed\n");

putchar('\n');

}

printf("For each species insert code and relative quantity.\nTo exit the loop enter 0 as a species code\n\n");

while (true) {

printf("\nInsert species code..................................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

args.species\_code = strtol(buffer\_for\_integer, NULL, 10);

if (args.species\_code == 0) {

if (args.pack\_number == 0) {

printf("A pack cannot be empty!\nOperation aborted\n");

goto exit\_with\_a\_failure;

} else {

break;

}

}

printf("Insert quantity......................................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

args.quantity = strtol(buffer\_for\_integer, NULL, 10);

if (args.pack\_number == 0) {

if (!attempt\_exec\_op\_on\_pack(&args, true)) {

printf("Operation failed\n");

goto exit\_with\_a\_failure;

}

} else {

if (!attempt\_exec\_op\_on\_pack(&args, false)) {

printf("Operation failed\n");

goto exit\_with\_a\_failure;

}

}

}

printf("New pack (no. %u) has been created for order %010u\n", args.pack\_number, args.order\_id);

exit\_with\_a\_failure:

putchar('\n');

printf("Press enter key to get back to menu ...\n");

getchar();

}

void run\_as\_order\_processor(char \*username)

{

struct configuration cnf;

char choice;

memset(&cnf, 0, sizeof(cnf));

memset(curr\_user, 0, sizeof(curr\_user));

strncpy(curr\_user, username, BUFFSIZE\_L);

if (parse\_config("config/opc.user", &cnf, "=")) {

fprintf(stderr, "Invalid configuration file selected (OPP)\n");

exit(EXIT\_FAILURE);

}

if (mysql\_change\_user(conn, cnf.username, cnf.password, cnf.database)) {

fprintf(stderr, "Unable to switch privileges\n");

exit(EXIT\_FAILURE);

}

while (true) {

init\_screen(true);

printf("Welcome %s\n\n", curr\_user);

printf("\*\*\* What do you wanna do? \*\*\*\n\n");

printf("1) Create new pack\n");

printf("2) View details about order processing\n");

printf("p) Change password\n");

printf("q) Quit\n");

choice = multi\_choice("Pick an option", "12pq", 4);

switch (choice) {

case '1': create\_pack(); break;

case '2': report\_packs(); break;

case 'p': change\_password(curr\_user); break;

case 'q': printf("Bye bye!\n\n\n"); return;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

}

}

FILE: parser.c

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

#include "defines.h"

#define LINE 1024

bool parse\_config(const char \*path, struct configuration \*conf, const char \* delimiter)

{

char line[LINE];

char \*key;

char \*value;

int no\_lines = 0;

FILE \*file = fopen(path, "r");

if (file == NULL)

return 1;

while (fgets(line, LINE, file) != NULL && no\_lines < 5) {

++ no\_lines;

key = strtok(line, delimiter);

value = strtok(NULL, delimiter);

value[strcspn(value, "\n")] = '\0'; // necessario per rimuovere il '\n' finale

if (strcmp(key, "host") == 0)

strncpy(conf->host, value, BUFFSIZE\_L);

else if (strcmp(key, "username") == 0)

strncpy(conf->username, value, BUFFSIZE\_L);

else if (strcmp(key, "password") == 0)

strncpy(conf->password, value, BUFFSIZE\_L);

else if (strcmp(key, "database") == 0)

strncpy(conf->database, value, BUFFSIZE\_L);

else if (strcmp(key, "port") == 0)

conf->port = strtol(value, NULL, 10);

else

return 1;

}

fclose(file);

return (no\_lines != 5);

}

FILE: utils.c

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

#include <mysql.h>

void init\_screen(bool title)

{

(title) ? system("clear && cat head") : system("clear");

}

void print\_error(MYSQL \*conn, char \*message)

{

fprintf(stderr, "%s\n", message);

if (conn != NULL) {

fprintf(stderr, "Error %u (%s): %s\n",

mysql\_errno (conn),

mysql\_sqlstate(conn),

mysql\_error (conn));

}

}

void print\_stmt\_error (MYSQL\_STMT \*stmt, char \*message)

{

fprintf (stderr, "%s\n", message);

if (stmt != NULL) {

fprintf(stderr, "Error %u (%s): %s\n",

mysql\_stmt\_errno (stmt),

mysql\_stmt\_sqlstate(stmt),

mysql\_stmt\_error (stmt));

}

}

bool setup\_prepared\_stmt(MYSQL\_STMT \*\*stmt, char \*statement, MYSQL \*conn)

{

my\_bool update\_length = true;

\*stmt = mysql\_stmt\_init(conn);

if (\*stmt == NULL) {

print\_error(conn, "Could not initialize statement handler");

return false;

}

if (mysql\_stmt\_prepare (\*stmt, statement, strlen(statement)) != 0) {

print\_stmt\_error(\*stmt, "Could not prepare statement");

return false;

}

mysql\_stmt\_attr\_set(\*stmt, STMT\_ATTR\_UPDATE\_MAX\_LENGTH, &update\_length);

return true;

}

static void print\_dashes(MYSQL\_RES \*res\_set)

{

MYSQL\_FIELD \*field;

unsigned int i, j;

mysql\_field\_seek(res\_set, 0);

putchar('+');

for (i = 0; i < mysql\_num\_fields(res\_set); i++) {

field = mysql\_fetch\_field(res\_set);

for (j = 0; j < field->max\_length + 2; j++)

putchar('-');

putchar('+');

}

putchar('\n');

}

static void dump\_result\_set\_header(MYSQL\_RES \*res\_set)

{

MYSQL\_FIELD \*field;

unsigned long col\_len;

unsigned int i;

/\* determine column display widths -- requires result set to be \*/

/\* generated with mysql\_store\_result(), not mysql\_use\_result() \*/

mysql\_field\_seek (res\_set, 0);

for (i = 0; i < mysql\_num\_fields (res\_set); ++i) {

field = mysql\_fetch\_field (res\_set);

col\_len = strlen(field->name);

if (col\_len < field->max\_length)

col\_len = field->max\_length;

if (col\_len < 4 && !IS\_NOT\_NULL(field->flags))

col\_len = 4; /\* 4 = length of the word "NULL" \*/

field->max\_length = col\_len; /\* reset column info \*/

}

print\_dashes(res\_set);

putchar('|');

mysql\_field\_seek (res\_set, 0);

for (i = 0; i < mysql\_num\_fields(res\_set); ++i) {

field = mysql\_fetch\_field(res\_set);

printf(" %-\*s |", (int)field->max\_length, field->name);

}

putchar('\n');

print\_dashes(res\_set);

}

bool dump\_result\_set(MYSQL\_STMT \*stmt, char \*title, int leading\_zeros\_bitmask)

{

int i;

int status;

int num\_fields; /\* number of columns in result \*/

MYSQL\_FIELD \*fields; /\* for result set metadata \*/

MYSQL\_BIND \*rs\_bind; /\* for output buffers \*/

MYSQL\_RES \*rs\_metadata;

MYSQL\_TIME \*date;

size\_t attr\_size;

/\* Prefetch the whole result set. This in conjunction with

\* STMT\_ATTR\_UPDATE\_MAX\_LENGTH set in `setup\_prepared\_stmt`

\* updates the result set metadata which are fetched in this

\* function, to allow to compute the actual max length of

\* the columns.

\*/

if (mysql\_stmt\_store\_result(stmt)) {

fprintf(stderr, " mysql\_stmt\_execute(), 1 failed\n");

fprintf(stderr, " %s\n", mysql\_stmt\_error(stmt));

return false;

}

/\* the column count is > 0 if there is a result set \*/

/\* 0 if the result is only the final status packet \*/

num\_fields = mysql\_stmt\_field\_count(stmt);

if (num\_fields > 0) {

/\* there is a result set to fetch \*/

printf("%s\n", title);

if((rs\_metadata = mysql\_stmt\_result\_metadata(stmt)) == NULL) {

print\_stmt\_error(stmt, "Unable to retrieve result metadata");

return false;

}

dump\_result\_set\_header(rs\_metadata);

fields = mysql\_fetch\_fields(rs\_metadata);

rs\_bind = (MYSQL\_BIND \*) malloc(sizeof (MYSQL\_BIND) \* num\_fields);

if (!rs\_bind) {

print\_stmt\_error(stmt, "Cannot allocate output buffers");

return false;

}

memset(rs\_bind, 0, sizeof (MYSQL\_BIND) \* num\_fields);

/\* set up and bind result set output buffers \*/

for (i = 0; i < num\_fields; ++i) {

// Properly size the parameter buffer

switch(fields[i].type) {

case MYSQL\_TYPE\_DATE:

case MYSQL\_TYPE\_TIMESTAMP:

case MYSQL\_TYPE\_DATETIME:

case MYSQL\_TYPE\_TIME:

attr\_size = sizeof(MYSQL\_TIME);

break;

case MYSQL\_TYPE\_FLOAT:

attr\_size = sizeof(float);

break;

case MYSQL\_TYPE\_DOUBLE:

attr\_size = sizeof(double);

break;

case MYSQL\_TYPE\_TINY:

attr\_size = sizeof(signed char);

break;

case MYSQL\_TYPE\_SHORT:

case MYSQL\_TYPE\_YEAR:

attr\_size = sizeof(short int);

break;

case MYSQL\_TYPE\_LONG:

case MYSQL\_TYPE\_INT24:

attr\_size = sizeof(int);

break;

case MYSQL\_TYPE\_LONGLONG:

attr\_size = sizeof(int);

break;

default:

attr\_size = fields[i].max\_length;

break;

}

// Setup the binding for the current parameter

rs\_bind[i].buffer\_type = fields[i].type;

rs\_bind[i].buffer = malloc(attr\_size + 1);

rs\_bind[i].buffer\_length = attr\_size + 1;

if(rs\_bind[i].buffer == NULL) {

print\_stmt\_error(stmt, "Cannot allocate output buffers");

return false;

}

}

if(mysql\_stmt\_bind\_result(stmt, rs\_bind)) {

print\_stmt\_error(stmt, "Unable to bind output parameters");

return false;

}

/\* fetch and display result set rows \*/

while (true) {

status = mysql\_stmt\_fetch(stmt);

if (status == 1 || status == MYSQL\_NO\_DATA)

break;

putchar('|');

for (i = 0; i < num\_fields; ++i) {

if (rs\_bind[i].is\_null\_value) {

printf (" %-\*s |", (int)fields[i].max\_length, "NULL");

continue;

}

switch (rs\_bind[i].buffer\_type) {

case MYSQL\_TYPE\_VAR\_STRING:

case MYSQL\_TYPE\_TIMESTAMP:

case MYSQL\_TYPE\_NEWDECIMAL:

printf(" %-\*s |", (int)fields[i].max\_length, (char\*)rs\_bind[i].buffer);

break;

case MYSQL\_TYPE\_DATE:

date = (MYSQL\_TIME \*)rs\_bind[i].buffer;

printf(" %d-%02d-%02d |", date->year, date->month, date->day);

break;

case MYSQL\_TYPE\_STRING:

printf(" %-\*s |", (int)fields[i].max\_length, (char \*)rs\_bind[i].buffer);

break;

case MYSQL\_TYPE\_FLOAT:

case MYSQL\_TYPE\_DOUBLE:

printf(" %.02f |", \*(float \*)rs\_bind[i].buffer);

break;

case MYSQL\_TYPE\_LONG:

case MYSQL\_TYPE\_SHORT:

case MYSQL\_TYPE\_TINY:

/\* ad ogni iterazione verifico se l'i-esimo

\* bit della maschera e' settato ad uno.

\* In tal caso la stampa prevede un padding di 0

\* iniziali altrimenti una semplice spaziatura

\* come tutte le altre colonne

\*/

if ((leading\_zeros\_bitmask & (1 << i)) == (1 << i))

printf(" %0\*d |", (int)fields[i].max\_length, \*(int \*)rs\_bind[i].buffer);

else

printf(" %-\*d |", (int)fields[i].max\_length, \*(int \*)rs\_bind[i].buffer);

break;

case MYSQL\_TYPE\_LONGLONG:

printf(" %-\*lld |", (int)fields[i].max\_length, \*(long long int \*)rs\_bind[i].buffer);

break;

default:

printf("ERROR: Unhandled type (%d)\n", rs\_bind[i].buffer\_type);

abort();

}

}

putchar('\n');

print\_dashes(rs\_metadata);

}

mysql\_free\_result(rs\_metadata); /\* free metadata \*/

/\* free output buffers \*/

for (i = 0; i < num\_fields; ++i)

free(rs\_bind[i].buffer);

free(rs\_bind);

}

return true;

}

FILE: warehouseclerk.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

#include <mysql.h>

#include "defines.h"

struct insert\_supplier\_sp\_params {

char fiscal\_code[BUFFSIZE\_XS];

char name[BUFFSIZE\_S];

unsigned int species\_code;

char address[BUFFSIZE\_M];

};

struct spec\_info {

unsigned int species\_code;

char species\_name[132]; // |NOME COMUNE| + |NOME LATINO| + 3 + 1

unsigned int stock;

};

static char curr\_user[BUFFSIZE\_L];

static bool attempt\_search\_suppliers(char \*name)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[1];

char prompt[BUFFSIZE\_L];

memset(param, 0, sizeof(param));

memset(prompt, 0, sizeof(prompt));

param[0].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_nome VARCHAR(32)

param[0].buffer = name;

param[0].buffer\_length = strlen(name);

if(!exec\_sp(&stmt, param, "call visualizza\_fornitori(?)"))

return false;

if (strlen(name) > 0)

snprintf(prompt, BUFFSIZE\_L, "\nSearch results for \'%s\':", name);

else

snprintf(prompt, BUFFSIZE\_L, "\nSearch results:");

if (!dump\_result\_set(stmt, prompt, LEADING\_ZERO\_BITMASK\_IDX\_0)) {

CLOSE\_AND\_RETURN(false, stmt);

}

mysql\_stmt\_close(stmt);

return true;

}

static bool attempt\_select\_available\_species(unsigned int sup\_code)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[1];

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_fornitore INT

param[0].buffer = &sup\_code;

param[0].buffer\_length = sizeof(sup\_code);

if(!exec\_sp(&stmt, param, "call visualizza\_specie\_disponibili(?)"))

return false;

if (!dump\_result\_set(stmt, "\nSearch results:", LEADING\_ZERO\_BITMASK\_IDX\_0)) {

CLOSE\_AND\_RETURN(false, stmt);

}

mysql\_stmt\_close(stmt);

return true;

}

static unsigned int attempt\_insert\_new\_supplier(struct insert\_supplier\_sp\_params \*input)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[5];

unsigned int code;

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_STRING; // IN var\_codice\_fiscale CHAR(16)

param[0].buffer = input->fiscal\_code;

param[0].buffer\_length = strlen(input->fiscal\_code);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_nome VARCHAR(32)

param[1].buffer = input->name;

param[1].buffer\_length = strlen(input->name);

param[2].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_specie INT

param[2].buffer = &(input->species\_code);

param[2].buffer\_length = sizeof(input->species\_code);

param[3].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_indirizzo VARCHAR(64)

param[3].buffer = input->address;

param[3].buffer\_length = strlen(input->address);

param[4].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_codice\_fornitore INT

param[4].buffer = &code;

param[4].buffer\_length = sizeof(code);

if(!exec\_sp(&stmt, param, "call inserisci\_fornitore(?, ?, ?, ?, ?)"))

return 0;

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // OUT var\_codice INT

param[0].buffer = &code;

param[0].buffer\_length = sizeof(code);

if (!fetch\_res\_sp(stmt, param))

return 0;

mysql\_stmt\_close(stmt);

return code;

}

static void insert\_new\_supplier(void)

{

struct insert\_supplier\_sp\_params params;

unsigned int supplier\_code;

char spec\_name[BUFFSIZE\_M];

char buffer\_for\_integer[BUFFSIZE\_XS];

memset(&params, 0, sizeof(params));

memset(spec\_name, 0, sizeof(spec\_name));

memset(buffer\_for\_integer, 0, sizeof(buffer\_for\_integer));

init\_screen(false);

printf("\*\*\* Insert a new supplier \*\*\*\n");

printf("Insert fiscal code...............................................: ");

get\_input(BUFFSIZE\_S, params.fiscal\_code, false, true);

printf("Insert first name................................................: ");

get\_input(BUFFSIZE\_S, params.name, false, true);

if (ask\_for\_tips("Do you wanna search species by name to find the right code", 0)) {

printf("\nInsert the name to filter on (default all).......................: ");

get\_input(BUFFSIZE\_M, spec\_name, false, false);

if (!attempt\_search\_species(false, spec\_name))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert species code..............................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

params.species\_code = strtol(buffer\_for\_integer, NULL, 10);

printf("Insert address...................................................: ");

get\_input(BUFFSIZE\_M, params.address, false, true);

putchar('\n');

supplier\_code = attempt\_insert\_new\_supplier(&params);

if (supplier\_code > 0)

printf("New supplier inserted (SCODE: %010u)\n", supplier\_code);

else

printf("Operation failed\n");

printf("Press enter key to get back to menu ...\n");

getchar();

}

static bool attempt\_add\_supply\_availability(unsigned int supplier\_code, unsigned int species\_code)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[2];

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_fornitore INT

param[0].buffer = &supplier\_code;

param[0].buffer\_length = sizeof(supplier\_code);

param[1].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_specie INT

param[1].buffer = &species\_code;

param[1].buffer\_length = sizeof(species\_code);

if(!exec\_sp(&stmt, param, "call aggiungi\_disponibilita\_fornitura(?, ?)"))

return false;

if (!dump\_result\_set(stmt, "\nUpdated list:", 0)) {

CLOSE\_AND\_RETURN(false, stmt);

}

mysql\_stmt\_close(stmt);

return true;

}

static void add\_supply\_availability(void)

{

char buffer\_for\_integer[BUFFSIZE\_XS];

char spec\_name[BUFFSIZE\_M];

char sup\_name[BUFFSIZE\_S];

unsigned int sup\_code;

unsigned int spec\_code;

memset(buffer\_for\_integer, 0, sizeof(buffer\_for\_integer));

memset(spec\_name, 0, sizeof(spec\_name));

memset(sup\_name, 0, sizeof(sup\_name));

init\_screen(false);

printf("\*\*\* Add supply availability \*\*\*\n");

if (ask\_for\_tips("Do you wanna see a list of available suppliers", 12)) {

printf("\nInsert the name to filter on (default all).......................: ");

get\_input(BUFFSIZE\_S, sup\_name, false, false);

if (!attempt\_search\_suppliers(sup\_name))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert supplier code.............................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

sup\_code = strtol(buffer\_for\_integer, NULL, 10);

if (ask\_for\_tips("Do you wanna search species by name to find the right code", 0)) {

printf("\nInsert the name to filter on (default all).......................: ");

get\_input(BUFFSIZE\_M, spec\_name, false, false);

if (!attempt\_search\_species(false, spec\_name))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert species code..............................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

spec\_code = strtol(buffer\_for\_integer, NULL, 10);

putchar('\n');

if (attempt\_add\_supply\_availability(sup\_code, spec\_code))

printf("Availability succesfully added\n");

else

printf("Operation failed\n");

printf("Press enter key to get back to menu ...\n");

getchar();

}

static bool attempt\_add\_supply\_request(unsigned int supplier\_code, unsigned int species\_code, unsigned int quantity)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[3];

int flags = LEADING\_ZERO\_BITMASK\_IDX\_0 | LEADING\_ZERO\_BITMASK\_IDX\_1;

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_fornitore INT

param[0].buffer = &supplier\_code;

param[0].buffer\_length = sizeof(supplier\_code);

param[1].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_specie INT

param[1].buffer = &species\_code;

param[1].buffer\_length = sizeof(species\_code);

param[2].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_quantita INT

param[2].buffer = &quantity;

param[2].buffer\_length = sizeof(quantity);

if (!exec\_sp(&stmt, param, "call inserisci\_richiesta\_fornitura(?, ?, ?)"))

return false;

if (!dump\_result\_set(stmt, "\nPendant request for selected species:", flags)) {

CLOSE\_AND\_RETURN(false, stmt);

}

mysql\_stmt\_close(stmt);

return true;

}

static void add\_supply\_request(void)

{

char buffer\_for\_integer[BUFFSIZE\_XS];

char sup\_name[BUFFSIZE\_S];

unsigned int sup\_code;

unsigned int spec\_code;

unsigned int quantity;

memset(buffer\_for\_integer, 0, sizeof(buffer\_for\_integer));

memset(sup\_name, 0, sizeof(sup\_name));

init\_screen(false);

printf("\*\*\* Add supply request \*\*\*\n");

if (ask\_for\_tips("Do you wanna see a list of available suppliers", 10)) {

printf("\nInsert the name to filter on (default all).....................: ");

get\_input(BUFFSIZE\_S, sup\_name, false, false);

if (!attempt\_search\_suppliers(sup\_name))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert supplier code...........................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

sup\_code = strtol(buffer\_for\_integer, NULL, 10);

if (ask\_for\_tips("Do you wanna see avaliable species for selected supplier", 0)) {

if (!attempt\_select\_available\_species(sup\_code))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert species code............................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

spec\_code = strtol(buffer\_for\_integer, NULL, 10);

printf("Insert relative quantity.......................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

quantity = strtol(buffer\_for\_integer, NULL, 10);

putchar('\n');

if (!attempt\_add\_supply\_request(sup\_code, spec\_code, quantity))

printf("Operation failed\n");

putchar('\n');

printf("Press enter key to get back to menu ...\n");

getchar();

}

static bool attempt\_report\_stock(unsigned int range)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[1];

int counter = 0;

char prompt[BUFFSIZE\_XL];

struct spec\_info info;

int status;

memset(param, 0, sizeof(param));

memset(&info, 0, sizeof(info));

memset(prompt, 0, sizeof(prompt));

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_range INT

param[0].buffer = &range;

param[0].buffer\_length = sizeof(range);

if(!exec\_sp(&stmt, param, "call report\_giacenza(?)"))

return false;

do {

if(conn->server\_status & SERVER\_PS\_OUT\_PARAMS)

goto next;

if (counter == 0)

snprintf(prompt, BUFFSIZE\_XL, "\n\*\*\* Species details \*\*\*");

else if (counter % 2 == 0 && counter != 0)

snprintf(prompt, BUFFSIZE\_XL, "\n\n\n\*\*\* Species details \*\*\*");

else

snprintf(prompt, BUFFSIZE\_XL, "\nList of the 5 most frequently chosen suppliers:");

if (!dump\_result\_set(stmt, prompt, LEADING\_ZERO\_BITMASK\_IDX\_0)) {

CLOSE\_AND\_RETURN(false, stmt);

}

++counter;

next:

status = mysql\_stmt\_next\_result(stmt);

if (status > 0) {

print\_stmt\_error(stmt, "Unexpected condition");

CLOSE\_AND\_RETURN(false, stmt);

}

} while (status == 0);

mysql\_stmt\_close(stmt);

return true;

}

static void report\_stock(void)

{

char buffer\_for\_integer[BUFFSIZE\_XS];

unsigned int range;

memset(buffer\_for\_integer, 0, sizeof(buffer\_for\_integer));

init\_screen(false);

printf("\*\*\* View details of the species to be supplied \*\*\*\n");

printf("How many species do you wanna see into the report: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

range = strtol(buffer\_for\_integer, NULL, 10);

putchar('\n');

if (!attempt\_report\_stock(range))

printf("Operation failed\n");

printf("\n\nPress enter key to get back to menu ...\n");

getchar();

}

static int attempt\_add\_address(unsigned int supplier\_code, char \*address)

{

MYSQL\_STMT \*stmt;

MYSQL\_BIND param[2];

memset(param, 0, sizeof(param));

param[0].buffer\_type = MYSQL\_TYPE\_LONG; // IN var\_fornitore INT

param[0].buffer = &supplier\_code;

param[0].buffer\_length = sizeof(supplier\_code);

param[1].buffer\_type = MYSQL\_TYPE\_VAR\_STRING; // IN var\_indirizzo VARCHAR(64)

param[1].buffer = address;

param[1].buffer\_length = strlen(address);

if(!exec\_sp(&stmt, param, "call aggiungi\_indirizzo\_fornitore(?, ?)"))

return false;

if (!dump\_result\_set(stmt, "\nUpdated addresses list:", 0)) {

CLOSE\_AND\_RETURN(false, stmt);

}

mysql\_stmt\_close(stmt);

return true;

}

static void add\_address(void)

{

unsigned int sup\_code;

char address[BUFFSIZE\_M];

char buffer\_for\_integer[BUFFSIZE\_XS];

char sup\_name[BUFFSIZE\_M];

memset(address, 0, sizeof(address));

memset(buffer\_for\_integer, 0, sizeof(buffer\_for\_integer));

memset(sup\_name, 0, sizeof(sup\_name));

init\_screen(false);

printf("\*\*\* Add a address for a supplier \*\*\*\n");

if (ask\_for\_tips("Do you wanna see a list of available suppliers", 0)) {

printf("\nInsert the name to filter on (default all).....................: ");

get\_input(BUFFSIZE\_S, sup\_name, false, false);

if (!attempt\_search\_suppliers(sup\_name))

printf("Operation failed\n");

putchar('\n');

}

printf("Insert supplier code.................................: ");

get\_input(BUFFSIZE\_XS, buffer\_for\_integer, false, true);

sup\_code = strtol(buffer\_for\_integer, NULL, 10);

printf("Insert address.......................................: ");

get\_input(BUFFSIZE\_S, address, false, true);

putchar('\n');

if (attempt\_add\_address(sup\_code, address))

printf("Address \"%s\" for %010u succesfully added\n", address, sup\_code);

else

printf("\nOperation failed\n");

printf("Press enter key to get back to menu ...\n");

getchar();

}

void run\_as\_warehouse\_clerk(char \*username)

{

struct configuration cnf;

char choice;

memset(&cnf, 0, sizeof(cnf));

memset(curr\_user, 0, sizeof(curr\_user));

strncpy(curr\_user, username, BUFFSIZE\_L);

if (parse\_config("config/whc.user", &cnf, "=")) {

fprintf(stderr, "Invalid configuration file selected (ADM)\n");

exit(EXIT\_FAILURE);

}

if(mysql\_change\_user(conn, cnf.username, cnf.password, cnf.database)) {

fprintf(stderr, "Unable to switch privileges\n");

exit(EXIT\_FAILURE);

}

while (true) {

init\_screen(true);

printf("Welcome %s\n\n", curr\_user);

printf("\*\*\* What do you wanna do? \*\*\*\n\n");

printf("1) Insert a new supplier\n");

printf("2) Add address for a supplier\n");

printf("3) Add supply availability\n");

printf("4) Add supply request\n");

printf("5) View details of the species to be supplied\n");

printf("p) Change password\n");

printf("q) Quit\n");

choice = multi\_choice("Pick an option", "12345pq", 8);

switch (choice)

{

case '1': insert\_new\_supplier(); break;

case '2': add\_address(); break;

case '3': add\_supply\_availability(); break;

case '4': add\_supply\_request(); break;

case '5': report\_stock(); break;

case 'p': change\_password(curr\_user); break;

case 'q': printf("Bye bye!\n\n\n"); return;

default:

fprintf(stderr, "Invalid condition at %s:%d\n", \_\_FILE\_\_, \_\_LINE\_\_);

abort();

}

}

}

1. Indicare con E le entità, con R le relazioni [↑](#footnote-ref-1)
2. PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna. [↑](#footnote-ref-2)
3. IDX = index, UQ = unique, FT = full text, PR = primary. [↑](#footnote-ref-3)