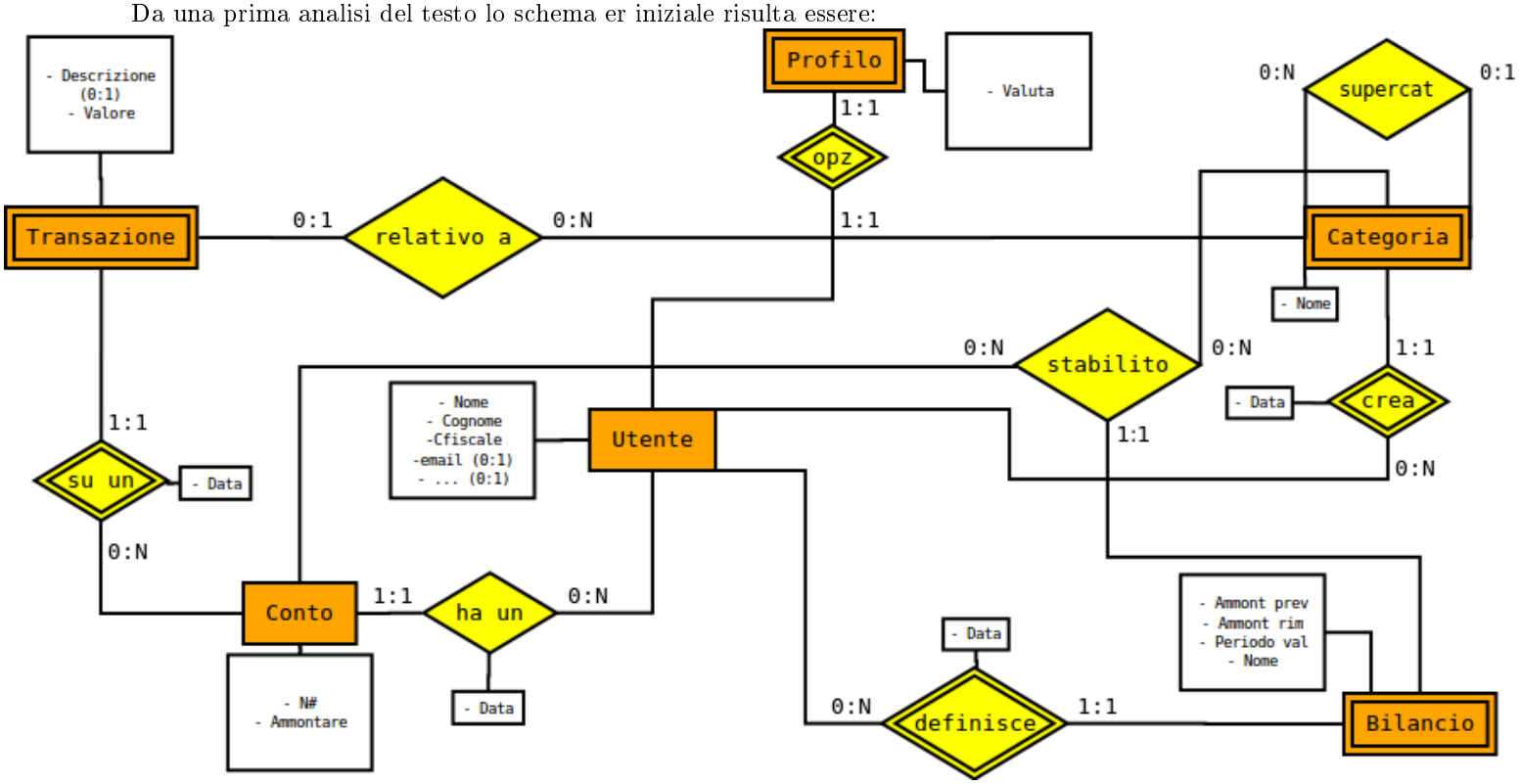
Progetto di Basi di Dati - Documentazione Tecnica

Michele Lazzeri | 822879

1 Progettazione Er

Da una prima analisi del testo lo schema er iniziale risulta essere:



Di queste sono considerate tipi di entità deboli (entità in cui la chiave è definita da un associazione con un altra entità):

- **Transazioni** la cui chiave è formata da un ID e dal numero del conto relativo, quest'ultimo derivante dall'associazione *Su_un*
- **Categoria** la cui chiave è formata dal nome della categoria e dall'identificatore dell'utente che l'ha creata, quest'ultimo derivante dall'associazione *crea*
- **Bilancio** la cui chiave è formata dal nome del bilancio e dall'identificatore dell'utente che l'ha definito, quest'ultimo derivante dall'associazione *definisce*
- **Profilo** la cui chiave è formata dall'identificatore dell'utente che l'ha definito, derivante dall'associazione *opz*

Per quanto riguarda le gerarchie abbiamo le seguenti suddivisioni:

- **Conto** viene suddivisa tramite gerarchia Totale Esclusiva in:
  - **Deposito**
  - **Credito**: Tetto_Massimo_Credito, Periodo Rinnovo, Data Iniziale
- **Transazioni** viene suddivisa tramite gerarchia Totale Esclusiva in:
  - **Spesa**
  - **Entrata**
- **Categoria** viene suddivisa tramite gerarchia Totale Esclusiva in :
  - **Categoria di spesa**
  - **Categoria di Entrata**

2 Ristrutturazione

Nella prima fase della ristrutturazione vengono analizzati i dati derivati: in questo caso:

- Ammontare di Conto e Bilancio vengono mantenuti anche se calcolabili a partire dalle spese: essendo la query 'Dammi il saldo del conto x' molto comune risulta molto più veloce l'accesso a tale informazione se contenuta direttamente in un'entità rispetto al calcolo di tale valore tramite somma di tutte le spese/entrate associate a tale entità.

Quindi vengono eliminate le gerarchie:

- **Conto**: viene riunita in una sola entita **Conto** in quanto spese, entrate e bilanci possono essere associate indistintamente a Conti di Credito o Conti di Debito. Viene quindi aggiunto l'attributo Tipo all'entità
- **Transazioni**: vengono mantenute solo le sottoclassi, in quanto i bilanci sono definiti solo per le spese
- **Categoria**: vengono mantenute le sottoclassi, in quanto non avrebbe senso suddividere le spese e le entrate e mantenere unite le relative categorie

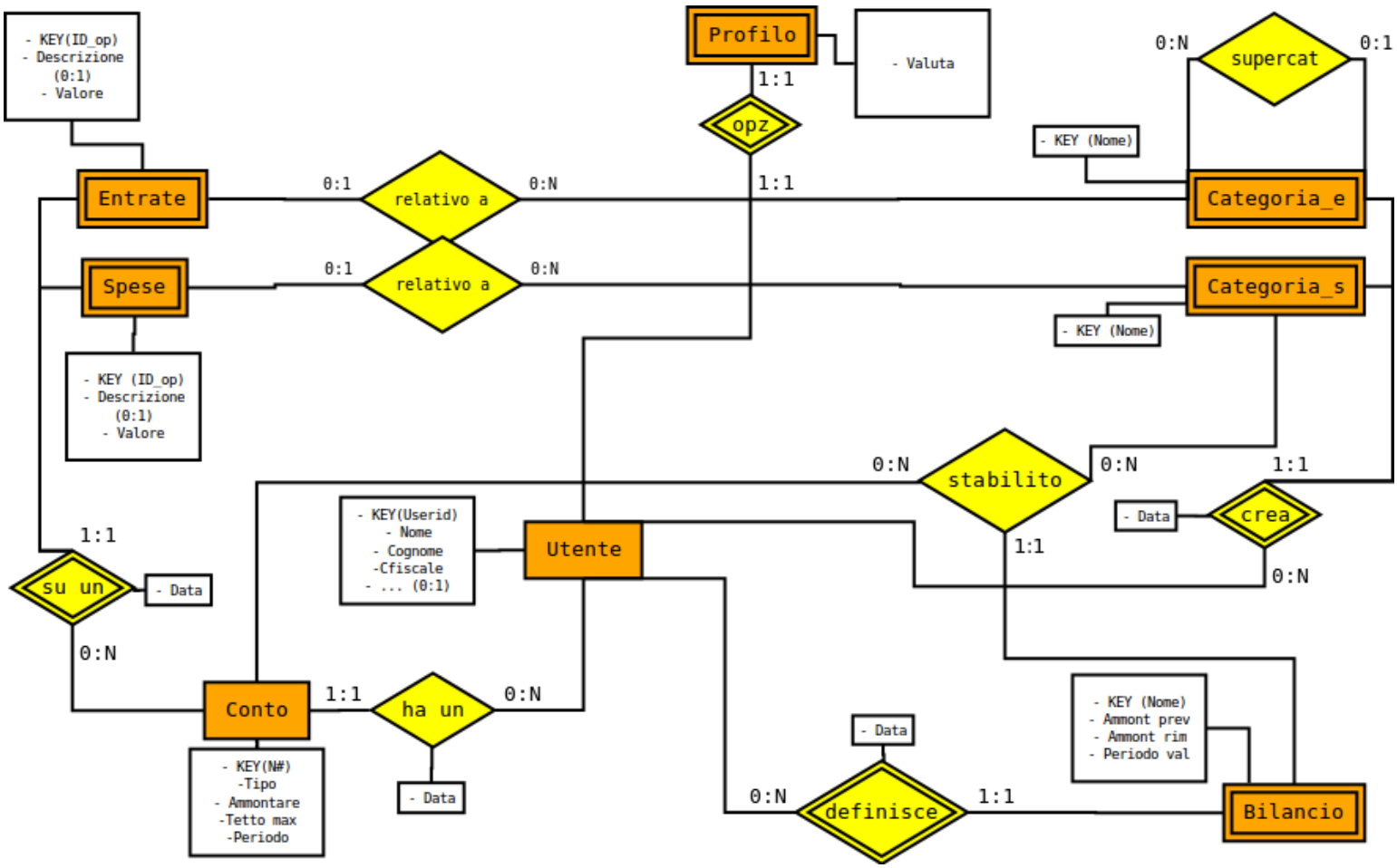
Quindi vengono definiti gli identificatori primari:

- **Conto**: Numero (creato perchè non esiste un insieme di attributi abbastanza ridotto da poter essere utilizzato in modo efficace)
- **Spesa**: Id_operazione e Conto.Numero
- **Entrata**: Id_operazione e Conto.Numero
- **Categoria**: Nome e Utente.ID
- **Bilancio**: Nome e Utente.ID
- **Utente**: Userid (creato perchè non esiste un insieme di attributi abbastanza ridotto da poter essere utilizzato in modo efficace)
- **Conto**: Utente.Userid

Note:

- l'idea di tenere suddivisi Profilo e Utente nonostante abbiano la stessa chiave è quella di suddividere le informazioni relative all'utente come Persona Fisica (dati "reali") e le informazioni utilizzate dall'applicazione

Quindi vengono eliminati gli attributi multivalore o composti: in questo schema non ne sono presenti
Alla fine della ristrutturazione lo schema ER si presenta come segue:



3 Traduzione

La traduzione da ER a Modello Relaziona porta al seguente schema relazionale:

CONTO (Numero, amm_disp, tipo, tettomax, scadenza_giorni, *utente*, datacreazione, *conto_di_rif*)

SPESA (*conto*, id_op, data, *cat_s_nome*, *cat_s_user*, descrizione, valore)

ENTRATA (*conto*, id_op, data, *cat_e_nome*, *cat_e_user*, descrizione, valore)

CATEGORIA_SPESA (*utente*, nome, supercat_nome)

CATEGORIA_ENTRATA (*utente*, nome, supercat_nome)

BILANCIO (*utente*, nome, ammontareprev, ammontarerest , periodovalidita, datapartenza)

UTENTE (userID, nome, cognome, email, indirizzo, cfiscale, citta, datadinascita, comunedinascita, nazioneresidenza, telefono)

PROFILO (*utente*, valuta)

BILANCIO_CONTO (*bilancio.utente, bilancio.nome, conto*)

BILANCIO_CAT (*bilancio.utente,bilancio.nome,categoria_s*)

Note:

- **CONTO**: viene aggiunto l'attributo *userid* derivante dall'associazione *ha un*, l'attributo data creazione derivante dall'associazione *ha un* e l'attributo *conto_di_rif* dall'associazione *riferito a*
- **SPESA**: viene aggiunto l'attributo *numeroconto* ed aggiunto alla chiave, dato il fatto che Spesa era un entità debole definitra tramite l'associazione *su un*
- **ENTRATA**: viene aggiunto l'attributo *numeroconto* ed aggiunto alla chiave, dato il fatto che Entrata era un entità debole definitra tramite l'associazione *su un*

- **Categoria spesa**: viene aggiunto l'attributo *userid* dato dall'associazione *crea*, l'attributo *supercat.nome* dato dall'associazione *supercat*. Non viene inserito anche l'attributo *supercat.userid* per ridondanza
- **Categoria entrata**: viene aggiunto l'attributo *userid* dato dall'associazione *crea*, l'attributo *supercat.nome* dato dall'associazione *supercat*. Non viene inserito anche l'attributo *supercat.userid* per ridondanza
- **Bilancio**: viene aggiunto l'attributo *userid* dato dall'associazione *definisce*
- **Profilo**: viene aggiunto l'attributo *userid* dato dall'associazione *opz*
- vengono aggiunte le relazioni **bilancio_conto** e **bilancio_cat** in quanto dato che *stabilito* è un'associazione molti-a-molti-a-molti non è possibile creare una sola tabella con Key(bilancio), Key(Conto), Key(categoria), perchè per ogni bilancio avrei più conti e più categorie, quindi ci si troverebbe a dovere togliere alcuni valori (es: Bilancio A, associato a conto 1,2,3 e cat c1,c2,c3. Le tuple della rel three-way sarebbero (A,1,c1),(A,2,c2),(A,3,c3), ma anche le spese del conto 2 nella categoria c1 dovrebbero rientrare in tale bilancio, ma in questo non rientrano) o inserirli tutti (esempio sopra: tuple: (A,1,c1),(A,1,c2),(A,1,c3),(A,2,c1),... sono troppe). Per cui creo due tabelle una per l'associazione Bilancio <-> Conto e una per Bilancio <-> Cat, cosicchè le tuple in B<->Conto diventano (A,1),(A,2),(A,3) e quelle B<->Cat (A,c1),(A,c2),(A,c3). Conto e Cat non necessitano di un'associazione, perchè ci sono operazioni della stessa cat su più conti e op dello stesso conto su più cat.

4 Normalizzazione

Considerando che lo schema non presenta attributi strutturati o molti valore si trova già in 1NF.
Inoltre ogni attributo non chiave dipende in funzionalmente e completamente dalla chiave primaria, quindi anche la 2NF è rispettata.
Per quanto riguarda la 3NF, ovvero la dipendenza non transitiva di ogni attributo non-chiave dalla chiave primaria, la relazione **Utente** con gli attributi datadinascita e comunedinascita non soddisfa tale forma. Infatti esistono le dipendenze cfiscale => datadinascita e la dipendenza cfiscale => comunedinascita e cfiscale non è una chiave. I due attributi possono quindi essere eliminati essendo calcolabili a partire dal cfiscale.
Lo schema è anche in BCNF in quanto non esistono dipendenze A->B in cui A non chiave determina B.

5 Schema Finale

Di ausilio alle funzioni dell'applicazione vengono inoltre create le seguenti relazioni: **nazione** con un solo attributo chiave name in cui vengono conservate le possibili nazioni **valuta** con un solo attributo chiave simbolo in cui vengono conservate le possibili valute
Vengono inoltre aggiunti alla relazione profilo un attributo username e password_hashed per contenere username e password criptata per la gestione dell'accesso all'applicazione.
Lo schema finale risulta quindi essere:

conto (Numero, amm_disp, tipo, tettomax, scadenza_giorni, *utente*, datacreazione, *conto_di_rif*)

spesa (*conto*, id_op, data, *cat_s_nome, cat_s_user*, descrizione, valore)

entrata (*conto*, id_op, data, *cat_e_nome, cat_e_user*, descrizione, valore)

categoria_spesa (*utente*, nome, supercat_nome)

categoria_entrata (*utente*, nome, supercat_nome)

bilancio (*utente*, nome, ammontareprev, ammontarerest , periodovalidita, datapartenza)

utente (userID, nome, cognome, email, indirizzo, cfiscale, citta, nazioneresidenza, telefono)

profilo (*utente*, valuta, username, password_hashed)

bilancio_conto (*bilancio.utente, bilancio.nome, conto*)

bilancio_cat (*bilancio.utente,bilancio.nome,categoria_s*)

nazione (name)

valuta (simbolo)

6 Prodotti Software Utilizzati

Per la realizzazione del progetto sono stati utilizzati:

- Apache come server web
- PostgreSQL come DBMS relazionale
- PHP 5 per la realizzazione delle pagine web dinamiche
- la libreria jpgraph di PHP per la visualizzazione dei grafici (http://jpgraph.net/)
- Javascript per alcune funzioni di visualizzazione
- lo script javascript jason's calendar (http://www.dynamicdrive.com/dynamicindex7/jasoncalendar.htm) per la generazione dei form di inserimento date
- CSS3 e HTML5 come linguaggi base per le pagine web
- PLPGSQL come linguaggio per la realizzazione delle funzioni in PostgreSQL

7 Descrizione funzioni

7.1 Funzioni PLPGSQL

Le funzioni sono presenti nei files src/prefunct.sql src/postfunct.sql e src/triggers.sql

- prefunct.sql Contiene le funzioni che vanno inserite prima delle tabelle
  - get_first_free_utente: cerca il massimo degli userid e restituisce massimo+1
  - get_first_free_conto: cerca il massimo dei numeri di conto e restituisce massimo+1
  - get_first_free_spentr: cerca il massimo delle spese e entrate e restituisce massimo+1
  - Queste tre funzioni sostituiscono l'uso del tipo SERIAL o delle sequenze, in quanto permettono di non avere troppi id vuoti. In pratica, utilizzando SERIAL o una sequenza, nel momento in cui avviene l'inserimento di una spesa tale sequenza viene portata a n+1. Se però l'inserimento non va a buon fine (a causa di vincoli o trigger) la sequenza non viene riportata a n, quindi n+1 rimane inutilizzato e al prossimo inserimento verrà usato n+2. Usando queste funzioni, invece, se l'inserimento dell'utente con id n+1 non va a buon fine, l'id n+1 verrà utilizzato per il prossimo inserimento.
- triggers.sql Contiene le funzioni fatte scattare dai trigger e la definizione di questi ultimi
  - tr_upd_spesa_id, la relativa funzione, tr_upd_entrata_id e la relativa funzione prendono la spesa/entrata inserita e modificano l'id chiamando la funzione get_first_free_spentr. Questo perchè la numerazione dipende dal conto, infatti la chiave delle relazioni spesa e entrata è conto,id_op . Quindi id_op deve prendere il valore n+1 dove n è il massimo dell'id per quel conto. Dato che sql non permette di passare direttamente tramite DEFAULT una funzione i cui parametri sono stabiliti nell'inserimento stesso, questa limitazione viene aggirata impostando a 0 l'id della spesa/entrata e creando poi questo trigger che calcola l'id corretto e lo modifica
  - tr_create_defaults e la relativa funzione all'inserimento di un utente creano le categorie di default e un profilo di default.
  - tr_check_referral_account e la relativa funzione controllano, all'inserimento di un conto di credito, che il conto di debito associato sia effettivamente di debito, che appartenga allo stesso utente e che la data di creazione del conto di credito non sia antecedente alla data di creazione del conto di debito
  - tr_initial_deposit e la relativa funzione creano le entrate iniziali dei conti di deposito e di credito
  - tr_check_date_spesa e la relativa funzione controllano che la data della spesa non sia antecedente alla creazione del conto e controlla anche che la categoria, se impostata, appartenga allo stesso utente
  - tr_check_date controlla che la data della creazione del bilancio non sia anteriore alla creazione del conto e che appartengano allo stesso utente
  - tr_set_default_amount imposta l'ammontare rimanente del bilancio uguale all'ammontare stabilito all'inserimento di un nuovo bilancio
  - tr_upd_account_on_spesa all'inserimento di una spesa aggiorna l'ammontare del conto associato e controlla che ci sia disponibilità sufficiente

  - tr_upd_account_on_entrata all'inserimento di una spesa aggiorna l'ammontare del conto e proibisce l'inserimento di entrate per i conti di credito (tranne che per l'entrate da rinnovo)
  - tr_upd_bilancio_on_spesa aggiorna l'ammontare disponibile dei bilanci associati a tale spesa

- postfunct.sql Contiene le funzioni che vanno inserite dopo le tabelle
  - fixall_til implementa la funzione di *testing* lanciata all'aggiornamento della data tramite form nell'applicazione. Questa funzione calcola per ogni conto di credito gli ammontare parziali di spesa per ogni periodo, controlla che non esista già una spesa sul relativo conto di riferimento e inserisce una spesa di Addebito da carta di credito. Ricalcola quindi l'ammontare dei conti di deposito e aggiorna l'ammontare disponibile dei bilanci. Spiegazione più dettagliata di tale funzione nel manuale utente alla sezione 2.
  - upd_fixall chiama la precedente funzione passando la data attuale come parametro
  - get_last_period_start_bil passando come parametri la chiave del bilancio e una data D restituisce la prima data precedente a D di rinnovo del bilancio
  - get_last_period_start_cred passando come parametri la chiave del conto di credito e una data D restituisce la prima data precedente a D di rinnovo del conto di credito
  - verifica_appartenenza restituisce 1 se il conto appartiene all'utente, 0 altrimenti
  - fix_cron ricevendo come parametro una data aggiorna calcola per i conti di credito in scadenza il totale delle spese del periodo, crea una spesa nei relativi conti di debito e aggiorna l'ammontare dei bilanci in scadenza

7.2 Funzioni PHP

Le principali funzioni PHP contenute nel file www/lib/lib_base.php usate sono:

- connection_pgsql che esegue la connessione al db tramite il file conf.php e restituisce la risorsa di connessione
- query che esegue le funzioni pg_prepare e pg_execute e restituisce la risorsa
- date_to_dmy che restituisce la data formattata d/m/Y
- decimal_to_currency che restituisce un importo formattato
- select_to_table a partire da una query formatta l'output in una tabella (descrizione più dettagliata nel file lib_base.php)
- select_to_tablewsum come la precedente ma inserisce una linea in cui calcola la somma di alcune colonne
- select_to_select_form crea un menu a tendina contenente il risultato di una query
- sanitize_number restituisce un numero nel formato XXXX.YY