Curs 1: Diagrama entitate relație – Entități și chei primare

Modelul entitate relație este un model conceptual de nivel înalt, neformalizat, pentru reprezentarea unui sistem din lumea reală, independent de platforma hardware și de instrumentele software utilizate pentru memorarea și manipularea datelor.

A fost propus în 1976 de către Peter Chen.

Diagrama ER este o reprezentare grafică a modelului ER, care poate fi cu ușurință transformat în model relațional. Modelul relațional prezintă organizarea datelor într-o bază de date de tip relațional, sub formă de tabele. De regulă modelul ERD și modelul relațional sunt utilizate pentru modelarea unor baze de date în care există o organizare fixată a datelor (schemă fixă), spre deosebire de baze de date NoSql care permit mai multă flexibilitate în structurarea informației.

Tipurile de elemente care se regăsesc în alcătuirea diagramelor ERD sunt prezentate în figura 1.

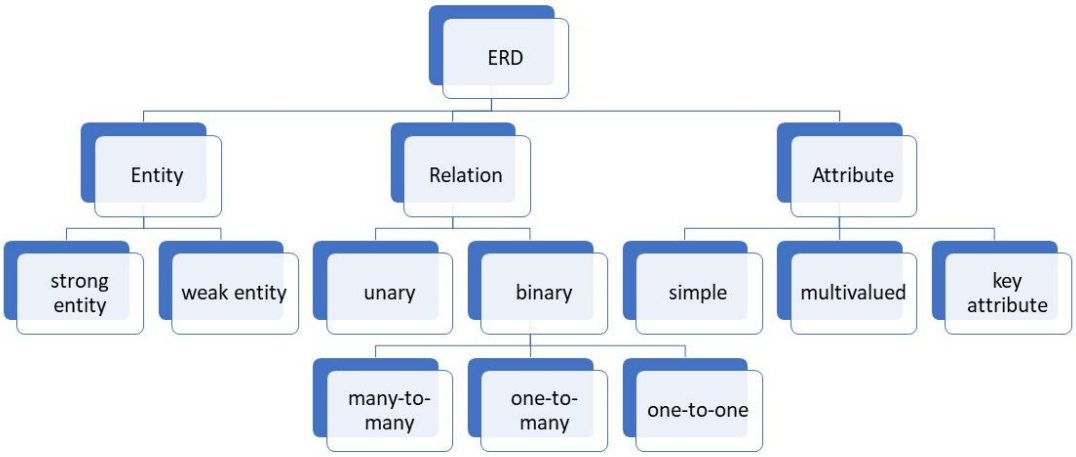


Figura 1: Componentele diagramei entitate relație

Entitate Entitățile desemnează persoane, locuri, evenimente, activități, obiecte semnificative etc. pentru sistemul pentru care se realizează modelul.

- De obicei substantiv;
- Devin tabele într-o bază de date relațională;
- Nu pot exista în aceeași diagramă două entități cu același nume.

Entitățile se reprezintă grafic sub formă de dreptunghiuri, numele unei entități se scrie în interiorul dreptunghiului cu majuscule.

Chei primare Cheia primară este un identificator unic în cadrul unei entității, făcând distincție între valori diferite ale acesteia. De regulă cheile primare se marchează cu simbolul #.

Cheia primară:

- trebuie să fie unică și cunoscută la orice moment, o entitate nu poate avea decât o singură cheie primară;
- trebuie să nu conțină informații descriptive, să fie simplă, fără ambiguități;
- să fie stabilă;
- să fie familiară utilizatorului.

Cum alegem cheia primară?

Este de remarcat faptul că nu avem o modalitate general aplicabilă tuturor entităților/tabelelor din baze de date pentru alegerea unei chei primare. O alegere potrivită pentru cheia primară se face în funcție de context.

Putem avea **chei simple** (un singur atribut/coloană) sau **chei compuse** (o grupare de atribute/coloane). Cheile compuse se adaugă, de obicei, la modelarea unei relații many-to-many ca tabel asociativ (a se vedea cursul despre transformarea diagramei entitate relație în model relațional) sau la modelarea unei entități dependente, a cărei existență depinde de existența unei alte entități.

O altă clasificare a cheilor primare se poate defini în funcție de semnificația pe care au raportat la celelalte atribute ale entității. Putem avea chei primare de tip ***natural key*** sau ***chei surogat***.

Natural keys sunt cheile formate cu atribute ale entității. Spre exemplu, fiecare produs are asociat un cod specific (business code), cu o anumită semnificație de business.

Cheile surogat sau sintetice/artificiale sunt alcătuite din coloane care nu reprezintă caracteristici specifice de business și care nu sunt corelate cu celelalte atribute, singurul rol fiind de a identifica unic liniile din tabelul care implementează entitatea. Spre exemplu, în funcție de SGBD putem avea chei generate cu secvențe, valori unice auto-incrementate, GUID-uri.

Avantajul folosirii cheilor naturale este că nu se va utiliza memorie suplimentară, folosind-se pentru identificarea liniilor date care deja există în tabel. În general, crearea unei constrângeri de cheie primară presupune crearea unui index care de asemenea, este consumator de spațiu. Pe de altă parte, este posibil ca o cheie surogat, spre exemplu codurile de business ale produselor, să nu respecte restricția de unicitate. De asemenea, codurile de business nu sunt întotdeauna stabile. Un produs poate fi etichetat de-a lungul timpului cu coduri distincte.

Un alt aspect important în alegerea unei chei primare este tipul de date utilizat, **numeric** sau **șir de caractere**. Este cunoscut faptul că operațiile cu tipuri numerice sunt mai rapide decât cele pe tipuri șir de caractere. De asemenea, este recomandat, în cazul unei chei primare de tip șir de caractere, ca dimensiunea să fie fixată. Șirurile cu lungime fixă oferă o performanță mai bună decât șirurile cu lungime variabilă.

Cheile primare de tip șir de caracter pot fi utilizate în situația în care același informații sunt înregistrate în baze de date diferite. Spre exemplu, un cod unic de asigurare medicală (în RO, acesta se numește CID) poate fi atributul prin care să fie corelate informații despre istoricul medical din mai multe baze de date. În situația în care două biblioteci ar dori să-și reunească colecțiile de cărți, în cazul în care cheile primare ar fi secvențe (chei artificiale), procesul ar fi complicat de faptul că același cod numeric s-ar putea regăsi în ambele baze de date, identificând cărți diferite. Un candidat mai bun pentru cheia primară ar fi codul ISBN.

Există un tip special de coduri de tip șir de caracter care pot reprezenta chei primare:

UUID (universal unique identifiers)

Un identificator UUID este o secvență de 128 biți pseudo-aleatoare. Deși probabilitatea ca două UUID-uri, generate în conformitate cu standardele să fie identice nu este zero, acesta este neglijabilă. Astfel, UUID reprezintă o modalitate de a obține coduri unice nu doar la nivelul unei singure baze de date, ci **global**, în orice sistem ar fi utilizate.

Există 5 versiuni de UUID. MySQL implementează primul tip de UUID. În această versiune, un UUID este obținut prin concatenarea adresei MAC de 48-bit a mașinii care generează UUID, cu o secvență *timestamp* de 60-bit, restul pozițiilor fiind utilizate pentru versiune. Microsoft Sql Server implementează versiunea standard 4, ca o secvență de biți aleatoare.

O cheie UUID de tip 1 poate furniza, așadar, informații despre momentul în care au fost generate înregistrările. În același timp un UUID, spre deosebire de o secvență nu dezvălui utilizatorului niciun indiciu despre volumul de date înregistrat și ar reduce riscul ca datele să fie modificate accidental.

În general valorile generate aleator nu sunt un candidat bun pentru o cheie primară, pentru că în majoritatea SGBD-urilor, cheilor primare li se asociază un index de tip cluster, memorat ca arbore (*B-tree*), care sortează liniile în table în funcție de valoarea cheilor primare. Intern, datele sunt organizate în blocuri, iar blocurile sunt organizate în pagini. Dacă valorile cheii primare sunt generate în ordine, înregistrări cu valori consecutive ale cheii primare se vor afla în aceeași pagină. Dacă valorile cheii primare sunt generate aleator și nu în ordine, pentru fiecare inserare a unei noi valori, va fi necesară încărcarea unei alte pagini în memorie, cheile fiind dispersate în arborele care memorează indexul. În acest caz și performanța indecșilor secundari este de asemenea afectată. Un index secundar, folosit pentru optimizarea căutărilor în funcție de valori ale unor coloane non-cheie, este reprezentat, de asemenea, de un tip special de arbore de căutare (B-tree), în care frunzele reprezintă valori ale cheilor primare.

O soluție de compromis, pentru a păstra anumite avantaje pe care le oferă cheile de tip UUID (în special unicitatea la nivel global, nu doar la nivelul unei singure baze de date) fără un impact major asupra performanței, ar fi generarea de chei UUID secvențiale. Se pot crea algoritmi care să genereze aleator doar o parte a UUID-ului, prima succesiune de biți fiind generată secvențial (a se vedea [10] pentru exemple).

Entități dependente O entitate dependentă este o entitate a cărei existență este condiționată de existența unei alte entități. TASK depinde de PROIECT. Nu putem defini task-uri dacă nu avem proiecte cărora să le asociem. Cheia primară a unei entități dependente include cheia primară a sursei (nr_proiect) şi cel puţin o descriere a entităţii dependente (nr_task). Practic, pentru o entitate dependentă nu putem defini o cheie primară. Este nevoie de cheia entității de care depinde pentru a identifica o entitate dependentă. Entitatea dependentă se desenează prin dreptunghiuri cu linii mai subțiri sau cu linii duble.

Subentități și superentități. O subentitate este o specializare a unei entități superentitate. O subentitate are toate atributele superentității în care este inclusă și în plus are alte atribute specifice. Subentitatea se desenează prin dreptunghiuri incluse în superentitate.

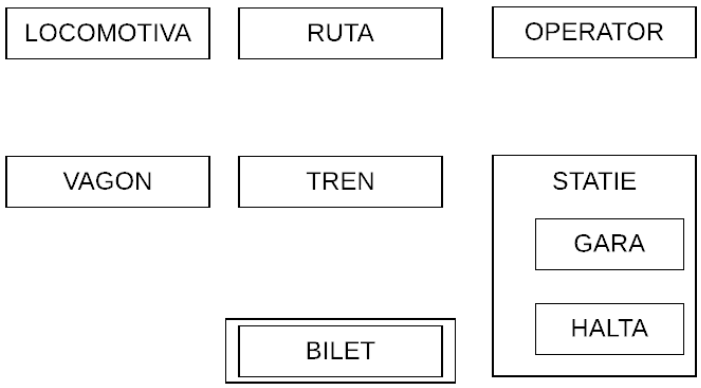
Pași în proiectarea diagramei entitate relație:

1. Stabilirea entităților din cadrul sistemului analizat;
2. Identificare relațiilor dintre entități și stabilirea cardinalităților;
3. Identificarea atributelor aferente entităților și relațiilor;
4. Identificarea cheilor.

EXEMPLUL 1: TRENURI

Operatorii de transport feroviar pentru transport călători doresc implementarea unei platforme online pentru *achiziționarea biletelor*. Fiecare operator administrează trenuri (vagoane și locomotive) pe care le pot repartiza pe rutele stabilite în funcție de infrastructura existentă: stații și linii interoperabile. Repartizarea trenurilor se face la date și ore stabilite. În funcție de sezon există perioade în care anumite trenuri sunt suspendate. Stațiile pot fi gări sau halte.

Pentru rezervarea unei călătorii, este obligatorie selectarea rutei, a datei de plecare, a stației de plecare și a stației de sosire. Orice rezervare presupune plătirea și emiterea unui bilet pe care este indicat vagonul și locul selectat.



BILET este entitate dependentă, depinde de entitatea TREN. GARA si HALTA sunt sub-entități.

Cheia pentru entitatea TREN poate fi un cod alfanumeric, spre exemplu completat cu valori ca ‘R-E 11631’.

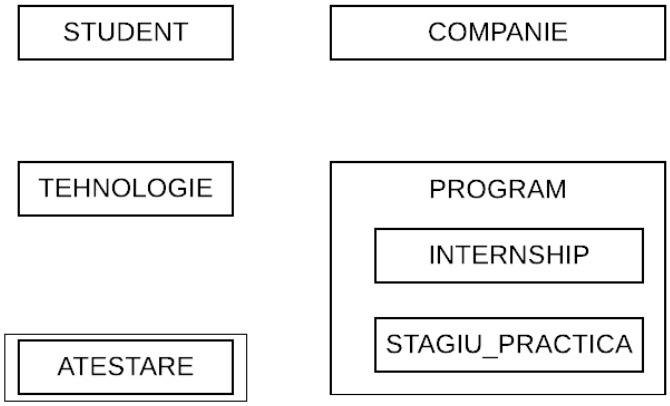
Deoarece o rezerva coincide cu eliberarea unui bilet, vom modela doar entitatea BILET, fără a modela entitatea REZERVARE.

EXEMPLUL 2: PROGRAME DE PREGĂTIRE

Studenții unei facultăți participă la programe de pregătire oferite de către companii. Programele pot fi de tip *internship* sau *stagiu de practică*. Programele de tip *internship* și stagiile de practică au o dată de început și o dată de final personalizate pentru fiecare student admis. Opțional, participarea la unele programe de *internship* poate fi plătită de către companii.

Pentru a fi admis la un program este necesar ca un student să cunoască unele tehnologii, la un anumit nivel cerut de companie.

La finalul programului, pentru fiecare student primește o atestare a finalizării programului.



INTERSHIP și STAGIU_PRACTICĂ sunt sub-entități. ATESTARE este entitate dependentă, depinde de STAGIU.

Nu vom modela entitatea FACULTATE, pentru că modelul se referă la studenții unei singure facultăți. Date despre facultate se pot memora la nivel de aplicație, ca simple variabile, nu este obligatoriu să fie memorate în baza de date.

Bibliografie

[1] Connolly T.M., Begg C.E., Database Systems: A Practical Approach to Design, Implementation and Management, 5th edition, Pearson Education, 2005

[2] Dollinger R., Andron L. – Baze de date şi gestiunea tranzacţiilor, Editura Albastră, Cluj-Napoca, 2004

[3] Oracle and/or its affiliates, Oracle Database Concepts, 1993, 2017

[4] Oracle and/or its affiliates, Oracle Database SQL Language Reference, 1996, 2017

[5] Oracle and/or its affiliates, Oracle Database Administrator's Guide, 2001, 2010

[6] Popescu I., Alecu A., Velcescu L., Florea (Mihai) G., Programare avansată în Oracle9i, Ed. Tehnică, 2004

[7] Popescu I., Velcescu L., Proiectarea bazelor de date, Ed. Universității din București, 2008

[8] Popescu I., Velcescu L., Neprocedural în Oracle 10g, Ed. Universității din București, 2008

[9] UUID https://en.wikipedia.org/wiki/Universally_unique_identifier

[10] ULID https://github.com/ulid/spec