TAMBook - Librărie online

1. Modul Analiză

### **Descrierea modelului ales și a obiectivelor aplicației**

TAMBook este o aplicație internă care îi permite administratorului să înregistreze o comandă pe care vrea să o plaseze utilizatorul care i-a cerut aceasta ce va conține cărțile alese în cantitatea dorită. De asemenea, dintre funcționalitățile aplicației se enumeră: vizualizarea comenzilor făcute de un utilizator ales, vizualizarea detaliilor pentru o comandă dată, vizualizarea cărților, editurilor, seriilor, categoriilor, autorilor, comenzilor și a detaliilor acestora, recenziilor, adreselor, orașelor, țărilor. Mai mult, aplicația prezintă o gestionare eficientă a produselor și stocurilor. Totodată aplicația oferă accesul la o serie de grafice dinamice pentru a se putea studia statisticile vânzărilor.

Astfel, un utilizator poate avea mai multe adrese care conțin date de livrare printre care și orașul care face parte dintr-o țară (o țară are mai multe orașe, iar un oraș poate fi inclus în mai multe adrese), iar acesta poate plasa mai multe comenzi la adrese diferite.

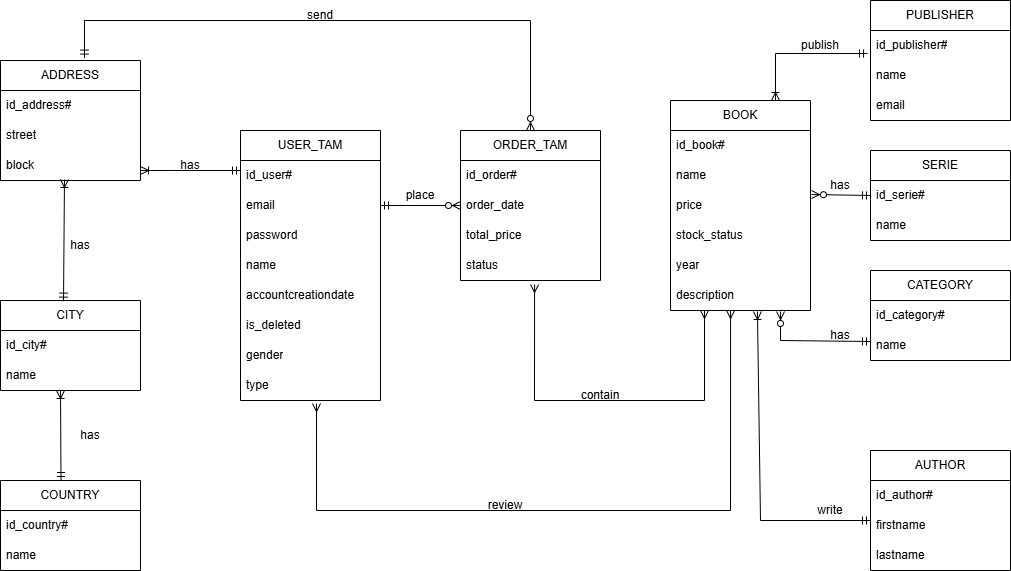
În ceea ce privește comanda, aceasta, inițial, se creează pentru un utilizator și adresa selectată, cu totalul pe 0, iar în timp ce sunt adăugate detaliile comenzii – cartea aleasă, cu prețul ei în momentul respectiv, cantitatea și discountul dacă este cazul – atunci totalul din comandă este actualizat pentru a corespunde cu realitatea.

Mai mult, în acest model, o carte este publicată de o singură editură, are o singură serie, o singură categorie și un singur autor, dar o serie, categorie, autor, editură pot avea mai multe cărți. Totodată, un utilizator poate scrie mai multe recenzii pentru cărțile dorite, chiar și mai multe în aceeași zi.

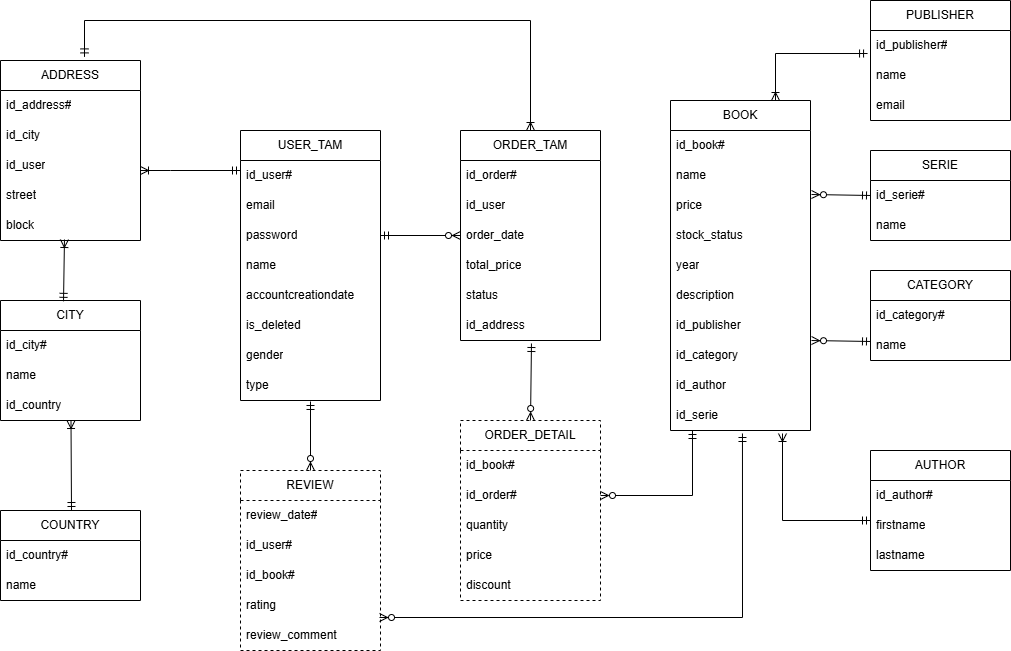
Aplicația are ca scop managementul unei librării online, prin gestionarea produselor și vânzărilor. Focusul principal al acesteia este pe creșterea vânzării produselor prin urmărirea tendințelor de cumpărare a acestora în funcție de mai mulți factori, precum: autor, serie, categorie și editura.

### **Diagrame**

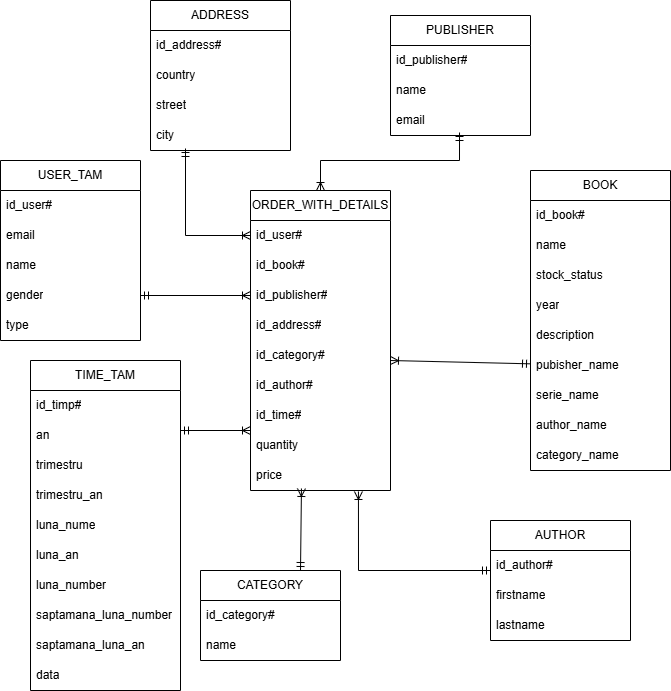
1. Diagrama Entitate-Relație a bazei de date OLTP



B. Diagrama Conceptuală a bazei de date OLTP



### **Diagrama stea a bazei de date depozit (un tabel de fapte și cel puțin 5 tabele dimensiune)**



### **Descrierea câmpurilor necesare pentru fiecare tabel din baza de date depozit și modul de populare al acestora cu informații din baza de date OLTP**

**Tabela faptă ORDER\_WITH\_DETAILS:**

Pentru a popula această tabelă, în schema OLTP s-a creat o vizualizare numită ORDER\_WITH\_DETAILS\_VW și care pentru a prelua datele corect a trebuit să facă două operații de join: între tabelele ORDER\_DETAIL și ORDER\_TAM prin atributul id\_order și între ORDER\_DETAIL și BOOK prin atributul id\_book, preluându-se în depozit doar înregistrările care au statusul din ORDER\_TAM marcate drept Completed (adică doar comenzile care au fost încheiate sunt duse în depozitul de date). Iar atributele vizualizării sunt preluate astfel din tabelele din OLTP: din ORDER\_TAM: id\_user, id\_address, order\_date, din ORDER\_DETAIL: id\_book, quantity, price, discount, din BOOK: id\_publisher, id\_category, id\_author. De asemenea, atributul price este calculat după formula: (od.price - (od.discount \* od.price) / 100) pentru ca în depozit prețul primit să aibă deja aplicat discountul. Astfel, utilizatorul din schema OLAP primește acces la această vizualizare și astfel poate insera datele.

Mai joi sunt câmpurile din tabela din baza de date depozit, iar explicațiile privind proveniența vor fi explicate în raport cu vizualizarea mai sus menționată:

* id\_user# - face parte din cheia primară și cheie externă, de tip number, care reprezintă id-ul unic al unui user și corespunde atributului id\_user din vizualizare.
* id\_book# - face parte din cheia primară și cheie externă, de tip number, care reprezintă id-ul unic al unei cărți și corespunde atributului id\_book din vizualizare.
* id\_publisher# - face parte din cheia primară și cheie externă, de tip number, care reprezintă id-ul unic al unei edituri și corespunde atributului id\_publisher din vizualizare.
* id\_address# - face parte din cheia primară și cheie externă, de tip number, care reprezintă id-ul unic al unei adrese și corespunde atributului id\_address din vizualizare.
* id\_category# - cheie primară și cheie externă, de tip number, care reprezintă id-ul unic al unei categorii și corespunde atributului id\_category din vizualizare.
* id\_author# - face parte din cheia primară și cheie externă, de tip number, care reprezintă id-ul unic al unui autor și corespunde atributului id\_book din vizualizare.
* id\_time# - face parte din cheia primară cheie primară și cheie externă, de tip date, care reprezintă id-ul unic al unei intrări temporale și corespunde atributului order\_date din vizualizare.
* quantity - tip NUMBER(10,0), care reprezintă numărul de bucăți cumpărate dintr-o carte și corespunde atributului quantity din vizualizare.
* price - tip NUMBER(10,2), care reprezintă prețul unei cărți la momentul plasării comenzii la care se aplică discountul și corespunde atributului price din vizualizare.

**Tabela ADDRESS:**

Pentru a popula această tabelă, în schema OLTP s-a creat o vizualizare numită ADDRESS\_VW și care pentru a prelua datele corect a trebuit să facă două operații de join: ADDRESS – CITY prin atributul id\_city și CITY – COUNTRY prin atributul id\_country. Iar atributele vizualizării sunt preluate astfel din tabelele din OLTP: din ADDRESS: id\_address, street, din COUNTRY: name care va fi salvat ca ‘country’, din CITY: name care va fi salvat ca ‘city’. Astfel, utilizatorul din schema OLAP primește acces la această vizualizare și astfel poate insera datele.

Mai joi sunt câmpurile din tabela din baza de date depozit, iar explicațiile privind proveniența vor fi explicate în raport cu vizualizarea mai sus menționată:

* id\_address# - cheie primară, de tip NUMBER, care reprezintă id-ul unic al unei adrese și corespunde atributului id\_address din vizualizare.
* country - tip VARCHAR(25), care reprezintă județul adresei unde se pot livra comenzi și corespunde atributului country din vizualizare.
* street - tip VARCHAR(25), care reprezintă strada adresei unde se pot livra comenzi și corespunde atributului street din vizualizare.
* city - tip VARCHAR(25), care reprezintă orașul adresei unde se pot livra comenzi și corespunde atributului city din vizualizare.

**Tabela PUBLISHER:**

Pentru a popula această tabelă, în schema OLTP s-a creat o vizualizare numită PUBLISHER\_VW și care pentru a prelua datele corect a trebuit să preia atributele id\_publisher, name și email din tabela PUBLISHER. Astfel, utilizatorul din schema OLAP primește acces la această vizualizare și astfel poate insera datele.

Mai joi sunt câmpurile din tabela din baza de date depozit, iar explicațiile privind proveniența vor fi explicate în raport cu vizualizarea mai sus menționată:

* id\_publisher# - cheie primară, de tip number, care reprezintă id-ul unic al unei edituri și corespunde atributului id\_publisher din vizualizare.
* name - tip VARCHAR(255), care reprezintă numele editurii și corespunde atributului name din vizualizare.
* email - tip VARCHAR(255), care reprezintă email-ul editurii și corespunde atributului email din vizualizare.

**Tabela BOOK:**

Pentru a popula această tabelă, în schema OLTP s-a creat o vizualizare numită BOOK\_VW și care pentru a prelua datele corect a trebuit să facă patru operații de join: între tabelele BOOK – PUBLISHER prin atributul id\_publisher, între BOOK – SERIE prin atributul id\_serie, între BOOK – AUTHOR prin atributul id\_author și între BOOK – CATEGORY prin atributul id\_category. Atributele vizualizării sunt preluate astfel din tabelele din OLTP: din BOOK: id\_book, name, stock\_status, year, description, din PUBLISHER: name care va fi salvat ca publisherName, din SERIE: name care va fi salvat ca serieName, din AUTHOR: se va concatena firstname cu lastname sub numele de authorName și din CATEGORY: name salvat precum categoryName. Astfel, utilizatorul din schema OLAP primește acces la această vizualizare și astfel poate insera datele.

Mai joi sunt câmpurile din tabela din baza de date depozit, iar explicațiile privind proveniența vor fi explicate în raport cu vizualizarea mai sus menționată:

* id\_book# - cheie primară, tip number, care reprezintă id-ul unic al unei cărți și corespunde atributului id\_book din vizualizare.
* name - tip VARCHAR(255), care reprezintă numele cărții și corespunde atributului name din vizualizare.
* stock\_status - de tip number(10,0), care reprezintă numărul de cărți disponibile în stoc și corespunde atributului stock\_status din vizualizare.
* year - tip NUMBER(5,0), care reprezintă anul în care a fost publicată cartea și corespunde atributului year din vizualizare.
* description - tip VARCHAR(255), care reprezintă sumarul cărții și corespunde atributului description din vizualizare.
* publisher\_name - tip VARCHAR(255), care reprezintă numele editurii cărții și corespunde atributului publisherName din vizualizare.
* serie\_name - tip VARCHAR(255), care reprezintă numele seriei cărții și corespunde atributului serieName din vizualizare.
* author\_name - tip VARCHAR(450), care reprezintă numele autorului cărții și corespunde atributului authorName din vizualizare.
* category\_name - tip VARCHAR(255), care reprezintă numele categoriei din care face parte cartea și corespunde atributului categoryName din vizualizare.

**Tabela AUTHOR:**

Pentru a popula această tabelă, în schema OLTP s-a creat o vizualizare numită AUTHOR\_VW și care pentru a prelua datele corect a trebuit să preia atributele id\_author, firstName, lastName din tabela AUTHOR. Astfel, utilizatorul din schema OLAP primește acces la această vizualizare și astfel poate insera datele.

Mai joi sunt câmpurile din tabela din baza de date depozit, iar explicațiile privind proveniența vor fi explicate în raport cu vizualizarea mai sus menționată:

* id\_author# - cheie primară, tip number, care reprezintă identificatorul unic pentru autorul respectiv și corespunde atributului id\_author din vizualizare.
* firstname - tip VARCHAR(255), care reprezintă prenumele autorului și corespunde atributului firstName din vizualizare.
* lastname - tip VARCHAR(255), care reprezintă numele de familie al autorului și corespunde atributului lastname din vizualizare.

**Tabela CATEGORY:**

Pentru a popula această tabelă, în schema OLTP s-a creat o vizualizare numită CATEGORY\_VW și care pentru a prelua datele corect a trebuit să preia atributele id\_category și name din tabela CATEGORY. Astfel, utilizatorul din schema OLAP primește acces la această vizualizare și astfel poate insera datele.

Mai joi sunt câmpurile din tabela din baza de date depozit, iar explicațiile privind proveniența vor fi explicate în raport cu vizualizarea mai sus menționată:

* id\_category# - cheie primară, de tip number, care reprezintă identificatorul unic al unei categorii și corespunde atributului id\_category din vizualizare.
* name - tip VARCHAR(255), care reprezintă denumirea unei categorii de cărți și corespunde atributului name din vizualizare.

**Tabela TIME\_TAM:**

* id\_timp# - cheie primară, de tip date, care este identificatorul unic al unei intrări temporale, reprezentat de data calendaristică din intervalul dorit.
* an - tip NUMBER(4), care reprezintă anul corespunzătoare intrării, obținut prin prelucrarea datelor calendaristice din sistem.
* trimestru - tip NUMBER(1), care reprezintă numărul trimestrului corespunzător intrării, obținut prin prelucrarea datelor calendaristice din sistem.
* trimestru\_an – tip VARCHAR2(10), care reprezintă trimestrul corespunzător intrării la care se concatenează anul intrării, obținut prin prelucrarea datelor calendaristice din sistem.
* luna\_nume – tip VARCHAR2(30), care reprezintă numele lunii corespunzătoare intrării, obținut prin prelucrarea datelor calendaristice din sistem.
* luna\_an – tip VARCHAR2(10), care reprezintă numărul lunii corespunzătoare intrării la care se concatenează anul intrării, obținut prin prelucrarea datelor calendaristice din sistem.
* luna\_number - tip NUMBER(2), care reprezintă numărul lunii corespunzătoare intrării, obținut prin prelucrarea datelor calendaristice din sistem.
* saptamana\_luna\_number – tip NUMBER(1), care reprezintă numărul săptămânii din lună corespunzătoare intrării, obținut prin prelucrarea datelor calendaristice din sistem.
* saptamana\_luna\_an – tip VARCHAR2(10), care reprezintă numărul săptămânii din lună corespunzătoare intrării la care se concatenează luna și anul intrării, obținut prin prelucrarea datelor calendaristice din sistem.
* data – tip DATE, care reprezintă data corespunzătoare intrării, obținut prin prelucrarea datelor calendaristice din sistem.

**Tabela USER\_TAM:**

Pentru a popula această tabelă, în schema OLTP s-a creat o vizualizare numită USER\_VW și care pentru a prelua datele corect a trebuit să preia atributele id\_user, email, name, gender și type din tabela USER\_TAM. Astfel, utilizatorul din schema OLAP primește acces la această vizualizare și astfel poate insera datele.

Mai joi sunt câmpurile din tabela din baza de date depozit, iar explicațiile privind proveniența vor fi explicate în raport cu vizualizarea mai sus menționată:

* id\_user# - cheie primară, de tip number, care reprezintă identificatorul unic al unui utilizator și corespunde atributului id\_user din vizualizare.
* email - tip VARCHAR(100), care reprezintă adresa de e-mail a utilizatorului și corespunde atributului email din vizualizare.
* name - tip VARCHAR(255), care reprezintă numele complet al utilizatorului și corespunde atributului name din vizualizare.
* gender - tip VARCHAR(5), care reprezintă genul utilizatorului (ex. "M", "F") și corespunde atributului gender din vizualizare.
* type - tip VARCHAR(20), care reprezintă tipul utilizatorului ("Persoana fizica" sau "Persoana juridica") și corespunde atributului type din vizualizare.

### **Identificarea constrângerilor specifice depozitelor de date ce trebuie definite, justificând alegerea făcută**

**Tabela faptă ORDER\_WITH\_DETAILS:**

* Cheie primară compusă id\_user, id\_book, id\_publisher, id\_address, id\_category, id\_author, id\_timp – DISABLE NOVALIDATE
* id\_user# - pentru foreign key: ENABLE NOVALIDATE
* id\_book# - pentru foreign key: ENABLE NOVALIDATE
* id\_publisher# - pentru foreign key: RELY DISABLE NOVALIDATE (am ales rely pt ca mai rar apare o editură nouă care să necesite să fie adăugată în schemă)
* id\_address# - pentru foreign key: ENABLE NOVALIDATE
* id\_category# - pentru foreign key: RELY DISABLE NOVALIDATE
* id\_author# - pentru foreign key: RELY DISABLE NOVALIDATE
* id\_time# - pentru foreign key: RELY DISABLE NOVALIDATE

Pentru atributele quantity și price nu se vor mai adăuga constrângeri suplimentare.

Constrângeri cheie primară compusă PK\_ORDER\_WITH\_DETAILS:

* S-a ales opțiunea de NOVALIDATE pentru că datele deja existente respectă condițiile constrângerii de cheie primară și nu se mai dorește o verificare a constrângerii pentru acestea.
* S-a ales opțiunea DISABLE pentru a permite încărcarea rapidă a datelor noi sau modificarea structurilor tabelei fără a afecta semnificativ performanța. Deoarece USER\_TAM este o tabelă de dimensiuni mici, impactul dezactivării indexului asociat cheii primare este minim asupra performanței interogărilor obișnuite.

1. Constrângerea de cheie externă fk\_order\_with\_details\_user

Constrângerea de cheie externă „fk\_order\_with\_details\_user” este definită cu opțiunile ENABLE ȘI NOVALIDATE, deoarece tabelei dimensiune USER\_TAM nu i se face update la fel de des precum tabela de fapte ORDER\_WITH\_DETAILS. O să existe cazuri în care se va observa o inconsistență a datelor în ceea ce privește corespondența dintre id\_user(fk) din ORDER\_WITH\_DETAILS și cel din tabela USER\_TAM (id\_user - pk).

S-a ales opțiunea de NOVALIDATE pentru că tabela ORDER\_WITH\_DETAILS fiind o tabelă de fapte, conține cele mai multe înregistrări, iar ca soluție de optimizare a preluării datelor este util ca sistemul să nu mai verifice dacă datele existente în tabele sunt valide relativ la restricțiile impuse de constrângere.

S-a ales opțiunea de ENABLE deoarece dorim să verificăm că actualizările și inserările noi respectă constrângerea.

2. Constrângerea de cheie externă fk\_order\_with\_details\_book

Constrângerea de cheie externă „fk\_order\_with\_details\_book” este definită cu opțiunile ENABLE ȘI NOVALIDATE, deoarece tabela dimensiune BOOK nu își face update la fel de des precum tabela de fapte ORDER\_WITH\_DETAILS. O să existe cazuri în care se va observa o inconsistență a datelor în ceea ce privește corespondența dintre id\_book(fk) din ORDER\_WITH\_DETAILS și cel din tabela BOOK(id\_book -pk).

S-a ales opțiunea de NOVALIDATE pentru că tabela ORDER\_WITH\_DETAILS fiind o tabelă de fapte, conține cele mai multe înregistrări, iar ca soluție de optimizare a preluării datelor este util ca sistemul să nu mai verifice dacă datele existente în tabele sunt valide relativ la restricțiile impuse de constrângere.

S-a ales opțiunea de ENABLE deoarece dorim să verificăm că actualizările și inserările noi respectă constrângerea.

3. Constrângerea de cheie externă fk\_order\_with\_details\_publisher

Constrângerea de cheie externă „fk\_order\_with\_details\_publisher” este definită cu opțiunile RELY DISABLE ȘI NOVALIDATE, deoarece nu sunt necesare verificări suplimentare, întrucât valorile atributului id\_publisher din tabela ORDER\_WITH\_DETAILS vor avea mereu corespondent în tabela dimensiune PUBLISHER prin atributul id\_publisher cu rol de cheie primară.

4. Constrângerea de cheie externă fk\_order\_with\_details\_address

Constrângerea de cheie externă „fk\_order\_with\_details\_address” este definită cu opțiunile ENABLE ȘI NOVALIDATE, deoarece tabela dimensiune ADDRESS nu își face update la fel de des precum tabela de fapte ORDER\_WITH\_DETAILS. O să existe cazuri în care se va observa o inconsistență a datelor în ceea ce privește corespondența dintre id\_address(fk) din ORDER\_WITH\_DETAILS și cel din tabela ADDRESS(id\_address -pk).

S-a ales opțiunea de NOVALIDATE pentru că tabela ORDER\_WITH\_DETAILS, fiind o tabelă de fapte, conține cele mai multe înregistrări, iar ca soluție de optimizare a preluării datelor este util ca sistemul să nu mai verifice dacă datele existente în tabele sunt valide relativ la restricțiile impuse de constrângere.

S-a ales opțiunea de ENABLE deoarece dorim să verificăm că actualizările și inserările noi respectă constrângerea.

5. Constrângerea de cheie externă fk\_order\_with\_details\_category

Constrângerea de cheie externă „fk\_order\_with\_details\_category” este definită cu opțiunile RELY DISABLE ȘI NOVALIDATE, valorile atributului id\_category din tabela ORDER\_WITH\_DETAILS vor corespunde întotdeauna valorilor atributului id\_ category din tabela CATEGORY, care are rol de cheie primară.

6. Constrângerea de cheie externă fk\_order\_with\_details\_author

Constrângerea de cheie externă „fk\_order\_with\_details\_author” este definită cu opțiunile RELY DISABLE ȘI NOVALIDATE, deoarece nu sunt necesare verificări suplimentare, întrucât valorile atributului id\_author din tabela ORDER\_WITH\_DETAILS vor avea mereu corespondent în tabela dimensiune AUTHOR prin atributul id\_author cu rol de cheie primară.

7. Constrângerea de cheie externă fk\_order\_with\_details\_time

Constrângerea de cheie externă „fk\_order\_with\_details\_time” este definită cu opțiunile RELY DISABLE ȘI NOVALIDATE, valorile atributului id\_time din tabela ORDER\_WITH\_DETAILS vor corespunde întotdeauna valorilor atributului id\_time din tabela TIME\_TAM, care are rol de cheie primară. Această asigurare provine din logica de preprocesare a datelor sau din constrângerile din sistemele sursă (se utilizează Sysdate), eliminând astfel necesitatea unor verificări suplimentare.

**Tabela ADDRESS:**

* id\_address# - cheie primară constrângere pk\_address: DISABLE VALIDATE
* S-a ales opțiunea de VALIDATE pentru a ne asigura că pentru datele deja existente sunt respectate condițiile constrângerii de cheie primară care presupune verificarea unicității și că valoarea nu este Null.
* S-a ales opțiunea de DISABLE pentru o încărcare mai rapidă a datelor noi, sau a modificărilor aduse tabelei. ADDRESS, fiind o tabelă de dimensiuni medii, alegerea de a nu se crea indexul nu scade prea mult performanța în momentul interogării pe baza cheii primare.

**Tabela PUBLISHER:**

* id\_publisher# - cheie primară constrângere pk\_publisher: DISABLE VALIDATE
* S-a ales opțiunea de VALIDATE pentru a ne asigura că pentru datele deja existente sunt respectate condițiile constrângerii de cheie primară care presupune verificarea unicității și că valoarea nu este Null.
* S-a ales opțiunea de DISABLE pentru o încărcare mai rapidă a datelor noi, sau a modificărilor aduse tabelei. Publisher, fiind o tabelă de dimensiuni mici, alegerea de a nu se crea indexul nu influențează foarte mult performanța în momentul interogării pe baza cheii primare.

**Tabela BOOK:**

* id\_book# - cheie primară constrângere pk\_book: DISABLE NOVALIDATE
* S-a ales opțiunea de NOVALIDATE pentru că datele deja existente respectată condițiile constrângerii de cheie primară și nu se mai dorește o verificare a constrângerii pentru acestea.
* S-a ales opțiunea DISABLE pentru a permite încărcarea rapidă a datelor noi sau modificarea structurilor tabelei fără a afecta semnificativ performanța. Deoarece tabela BOOK este o tabelă de dimensiuni nu prea mari și cheile primare sunt, în general, gestionate corect în procesul ETL (extracție, transformare, încărcare), impactul dezactivării indexului asociat este minim pentru interogările obținute.

**Tabela AUTHOR:**

* id\_author# - cheie primară constrângere pk\_author: DISABLE VALIDATE;
* S-a ales opțiunea de VALIDATE pentru a ne asigura că pentru datele deja existente sunt respectate condițiile constrângerii de cheie primară care presupune verificarea unicității și că valoarea nu este Null.
* S-a ales opțiunea de DISABLE pentru o încărcare mai rapidă a datelor noi, sau a modificărilor aduse tabelei. Author, fiind o tabelă de dimensiuni mici, alegerea de a nu se crea indexul nu influențează foarte mult performanța în momentul interogării pe baza cheii primare.

**Tabela CATEGORY:**

* id\_category# - cheie primară constrângere pk\_category: DISABLE VALIDATE
* S-a ales opțiunea de VALIDATE pentru a ne asigura că pentru datele deja existente sunt respectate condițiile constrângerii de cheie primară care presupune verificarea unicității și că valoarea nu este Null.
* S-a ales opțiunea DISABLE pentru a permite încărcarea rapidă a datelor noi sau modificarea structurilor tabelei fără a afecta semnificativ performanța .Deoarece CATEGORY este o tabelă de dimensiuni mici, impactul dezactivării indexului asociat cheii primare este minim asupra performanței interogărilor obișnuite.

**Tabela TIME\_TAM:**

* id\_time# - cheie primară constrângere pk\_time: DISABLE VALIDATE
* S-a ales opțiunea de VALIDATE pentru a ne asigura că pentru datele deja existente sunt respectate condițiile constrângerii de cheie primară care presupune verificarea unicității și că valoarea nu este Null.
* S-a ales opțiunea DISABLE pentru a permite încărcarea rapidă a datelor noi sau modificarea structurilor tabelei fără a afecta semnificativ performanța. Deoarece TIME\_TAM impactul dezactivării indexului asociat cheii primare este minim asupra performanței interogărilor obișnuite.

**Tabela USER\_TAM:**

* idUser# - cheie primară constrângere pk\_user: DISABLE NOVALIDATE
* S-a ales opțiunea de NOVALIDATE pentru că datele deja existente respectată condițiile constrângerii de cheie primară și nu se mai dorește o verificare a constrângerii pentru acestea.
* S-a ales opțiunea DISABLE pentru a permite încărcarea rapidă a datelor noi sau modificarea structurilor tabelei fără a afecta semnificativ performanța. Deoarece USER\_TAM este o tabelă de dimensiune nu prea mare, impactul dezactivării indexului asociat cheii primare este minim asupra performanței interogărilor obișnuite.

### **Identificarea indecșilor specifici depozitelor de date ce trebuie definiți asupra modelului; formularea unei cereri în limbaj natural care va determina utilizarea indecșilor specificați și va fi implementată în următoarea etapă**

Indecșii posibili pentru acest depozit de date pot fi reprezentați de coloanele gender (cu două valori: ‘M’, ‘F’) și type (cu două valori: ‘Persoana fizica’, ‘Persoana juridica’) din tabela USER\_TAM.

Dintre cei doi indecși posibili se va alege crearea indexului pe coloana type a tabelei USER\_TAM, tipul de index care va fi folosit fiind bitmap, permis deoarece coloana are doar două valori posibile. S-a ales acest index, pentru a optimiza timpul de returnare a informațiilor necesare când suntem interesați de tipul cumpărătorului care a plasat comanda.

Cerere: Să se afle care categorie de cărți a generat cele mai multe vânzări în rândul persoanelor fizice în anul trecut.

### **Identificarea obiectelor de tip dimensiune ce trebuie definite asupra modelului**

Actualul depozit de date permite crearea a două obiecte de tip dimensiune, pornind de la tabelele dimensiune din schema stea, TIME\_TAM și ADDRESS.

Pașii pentru crearea obiectului de tip dimensiune ADDRESS

1. Reamintirea diagramei conceptuale

În diagrama conceptuală, tabelele COUNTRY, CITY și ADDRESS, în această ordine sunt legate prin două relații de tipul “one to many”.

1. Proiectarea schemei stea

În schema stea, este creată tabela dimensiune ADDRESS, care cuprinde date din cele 3 tabele menționate mai sus: id\_address, country, city și street. Se observă că tabela dimensiune ADDRESS din schema stea este denormalizată.

1. Selectarea nivelurilor pentru obiectul de tip dimensiune

Atributele id\_address și street fac parte din vechiul tabel ADDRESS din OLTP, iar pentru niveluri se va alege id\_address. Astfel avem:

* LEVEL address IS (address.id\_address)
* LEVEL country IS (address.country)
* LEVEL city IS (address.city)

1. Stabilirea ierarhiei

address -> city -> country

Astfel avem:

HIERARCHY h\_address (

address CHILD OF

city CHILD OF

country );

1. Stabilirea dependențelor unidirecționale între atributele tabelei dimensiune ADDRESS – nu are
2. Comanda completa

CREATE DIMENSION address\_dim

LEVEL address IS (address.id\_address)

LEVEL country IS (address.country)

LEVEL city IS (address.city)

HIERARCHY h\_address (

address CHILD OF

city CHILD OF

country );

De asemenea, s-a mai creat și dimensiunea timp cu trei nivele, iar în ceea ce privește ierarhia s-au ales atributele an, trimestru\_an care reprezintă trimestrul dintr-un anume an și luna\_an care reprezinta luna dintr-un anumit an.

Astfel, un an are mai multe trimestre\_an care la rândul lor au doar un părinte an; un trimestru\_an are mai multe luna\_an, dar o luna\_an are un singur părinte trimestru\_an.

Comanda completă:

CREATE DIMENSION timp\_dim

LEVEL luna IS (time\_tam.luna\_an)

LEVEL trimestru IS (time\_tam.trimestru\_an)

LEVEL an IS (time\_tam.an)

HIERARCHY h\_timp(

luna CHILD OF

trimestru CHILD OF

an);

### **8. Identificarea tabelelor care vor fi partiționate și a tipului de partiționare; formularea unei cereri în limbaj natural care va determina utilizarea lor și va fi implementată în următoarea etapă**

Tabelul ORDER\_WITH\_DETAILS va fi partiționat în funcție de timp per trimestru (tabel TIME\_TAM), folosindu-se partiționarea prin ordonare. S-a ales acest tip de partiționare pentru că datele conținute sunt într-un interval istoric.

Cererea:

Să se afișeze detalii despre topul celor mai vândute 2 cărți per editură în primul trimestru al anului 2024 (primele 3 luni). Afișați numele, anul, categoria, descrierea cărții, numele editurii și al autorului.

### 9. Formularea în limbaj natural a unei cereri SQL complexe care va fi optimizată în următoarea etapă, folosind tehnici specifice bazelor de date depozit. Precizarea tehnicilor de optimizare ce ar putea fi utilizate pentru această cerere particulară (avantaje / dezavantaje de utilizare pentru o anumită tehnică)

Obțineți cei mai bine vânduți 4 autori (să se afișeze preț \* cantitate) per orașele: București, Barcelona, Milano, Lyon, Munchen, Mumbai și per trimestru, astfel:

* valoarea totală a vânzărilor pentru fiecare dintre cei mai bine 4 vânduți autori la nivel de oraș și trimestru.
* valoarea totală a vânzărilor pentru fiecare dintre cei mai bine 4 vânduți autori la nivel de oraș indiferent de trimestru.
* valoarea totală a vânzărilor pentru fiecare dintre cei mai bine 4 vânduți autori indiferent de oraș și trimestru.
* valoarea totală a vânzărilor indiferent de acest tip de autori, oraș și trimestru

Se doreste si vizualizarea unui clasament in functie de autori, oras, trimestru, dar si un clasament doar per oras cand valoarea totala a vanzarilor este pe toate nivelele, in rest se va face clasarea doar dupa valoarea totala. La aceasta situatie sa se adauge si un clasament per procente. Mai mult, sa fie eliminate liniile duplicat.

În implementare s-au folosit pentru complexitate:

* group by cu rollup
* grouping\_id
* grouping
* group\_id - pentru eliminare duplicate
* dense\_rank
* row\_number
* percent\_rank
* clauza WITH

Tehnicile de optimizare posibile: folosirea view-urilor materializate, funcții SQL specifice pentru grupare, indexi de tip join pe foreign key și folosirea tabelei order\_with\_details partiționată.

1. Vizualizări materializate

avantaj:

* eliminarea costurilor operațiilor join
* acces rapid la datele preagregate
* ideal pentru rezolvarea cerințelor care includ clasamente

dezavantaj:

* necesită spațiu în plus pentru stocarea datelor

1. Funcții SQL specifice pentru grupare (GROUPING, GROUPING\_ID, ROLLUP, PERCENT\_RANK,...)

avantaj:

* realizarea agregărilor la diferite niveluri de granularitate într-o singură interogare - ROLLUP
* existența funcțiilor de clasament - ROW\_NUMBER, DENSE\_RANK, PERCENT\_RANK

dezavantaj:

* combinarea multiplelor astfel de funcții poate îngreuna interpretarea și menținerea codului

1. Indecsi de tip join

avantaj:

* reducerea timpului de execuție pentru alăturarea tabelelor
* identificarea rapidă a datelor din tabelele din join

dezavantaj:

* necesită spațiu în plus în baza de date

1. Folosirea tabelei order\_with\_details partiționat

avantaj:

* reducerea timpului prin procesarea doar a partițiilor relevante

dezavantaj:

* performanța depinde de alegerea corectă a datelor cât să fie cuprinse într-o anume partiționare.

### 10. Formularea în limbaj natural a cel puțin 5 cereri, specifice DW, cu grad de complexitate diferit, concretizate în rapoarte (grafice) ce vor fi create în următoarele etape.

Cereri:

1. Afișați cele mai vândute 3 categorii dintr-o perioadă specificată, pentru cărțile care nu au rămas fără stoc. (pentru reprezentarea grafică se vor alege două date calendaristice)
2. Obțineți pentru cele mai cumpărate 5 edituri, procentele vânzărilor către persoane fizice și persoane juridice din totalul vânzărilor
3. Calculați pentru fiecare autor ale cărui cărți au fost comandate, poziția procentuală a valorii vânzărilor lunare a cărților care nu au stoc epuizat.
4. Comparați suma vânzărilor pentru toate editurile, din trimestrul curent, trimestrul anterior și cel care urmează raportat atât la trimestrele cu vânzări și la cele fără vânzări. În cererea SQL s-a selectat anul 2024 și toate trimestrele, iar pentru reprezentarea grafică se vor alege un număr de trimestru și un an.
5. Calculați valoarea vânzărilor dintr-o zi, totalul vânzărilor de la începutul lunii până la ziua selectată, diferența dintre vânzările din ziua curentă selectată și cele din ziua precedentă, media vânzărilor zilnice pe ultimele trei zile raportat la ziua selectată. În cererea SQL se va alege o zi din anul 2024, iar pentru raport se va introduce o dată pe baza căreia se va face statistica.