# Modelare practică a datelor

## Exemplu - Construirea unui model de date tradițional

Iată un exemplu de schiță[1] de construire a unui model de date tradițional. Este demn de remarcat faptul că, în acest exemplu, îl fac mai „realistă" decât alte exemple de modelare a datelor pe care le-am văzut. Majoritatea exemplelor presupun că porniți o afacere greenfield și construiți modelul de date de la zero. Practicienii constată că acest lucru se întâmplă rar. Realitatea este că majoritatea organizațiilor au un sistem de date care funcționează deja. În aceste cazuri, modelarea datelor este un act de înțelegere a ceea ce există deja. Indiferent dacă utilizați inginerie directă sau inversă, se aplică straturile conceptuale, logice și fizice ale modelării datelor.

Știați că, potrivit Gartner, 80% dintre proiectele legate de produse de date nu îndeplinesc așteptările. O cauză principală este lipsa unei înțelegeri comune între IT și departamentele de business.

Iată un exemplu de construire a unui model de date tradițional.

## Proiectarea Modelului Conceptual – Perspectiva afacerii

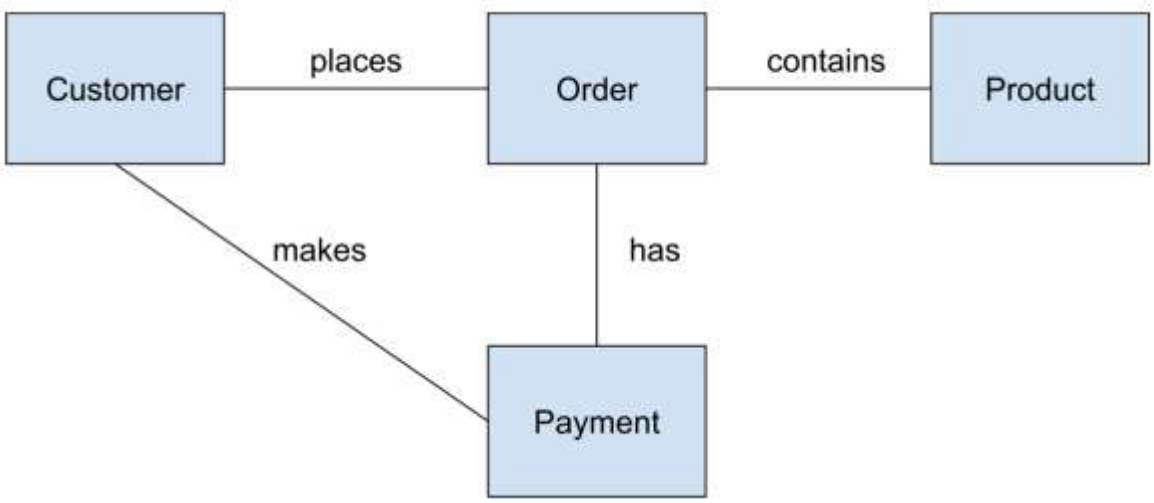
Modelarea conceptuală se referă la colaborarea cu părțile interesate pentru a înțelege lumea lor și a utiliza aceste interacțiuni pentru a crea un model de date la nivel înalt. Imaginați-vă că tocmai v-ați alăturat companiei noastre fictive de comerț electronic ca inginer de date. Nu există un model conceptual oficial (ar putea exista în mintea oamenilor) și ***vi s-a dat sarcina de a construi un model de date, de la conceptual la fizic.*** Scopul dvs. ar putea fi să utilizați acest model de date pentru a înțelege cum să ingerați, să transformați și să serviți date pentru analiză sau ML/AI. V-ați putea întreba: „Nu ne-am uitat tocmai la *construirea* unui Plan al Orașului Datelor, care presupune că nu există un oraș existent?" Și totuși ne-am implicat și, am făcut-o.

Deși multe exemple de modelare conceptuală a datelor presupun că pornești de la zero și construiești un model de date greenfield, acest lucru este mult mai puțin frecvent în realitate. De cele mai multe ori, te alături unei companii cu sisteme de aplicații existente, fără un model conceptual sau logic corespunzător. Realitatea este că majoritatea modelelor de date se produc organic, fără nicio planificare. Rezultatele sunt cele pe care le-ai putea aștepta. Calitatea datelor este haotică, iar modelele de date slabe afectează performanța bazei de date. Din experiența mea, excepția este atunci când un model de date este creat de la început, înainte de implementarea bazei de date fizice și încărcarea acesteia cu date. Aici, începem cu o implementare preexistentă a unui model de date. Aceleași reguli se aplică și lucrărilor de modelare conceptuală, indiferent dacă faci inginerie inversă sau inginerie directă. Scopul tău este să traduci limbajul afacerii într-un model de imagine de ansamblu care să o reprezinte.

Să folosim un exemplu extrem de simplu în care intervievez o parte interesată din afaceri, de exemplu pe Dana, șefa departamentului de Operațiuni de Comerț Electronic. Încerc să înțeleg un proces de afaceri, iar scopul meu este să traduc acest lucru într-un model conceptual.

**Eu:** ***Dana, pentru început, aș vrea să înțeleg cum percepi aspectele esențiale ale operațiunilor de comerț electronic. Cu cine sau cu ce te ocupi cel mai mult?***

**Dana:** Sigur. Fluxul nostru de lucru pentru comenzi este evident esențial pentru capacitatea noastră de a funcționa ca o companie online. Avem clienți, desigur. Aceștia plasează comenzi care conțin unul sau mai multe produse. De asemenea, urmărim plățile pentru aceste comenzi.
**Eu:** ***Grozav. Deci, doar ca să repet: lucrăm cu clienți, comenzi, produse și plăți. Cum sunt acestea legate?***
**Dana:** Un client poate plasa mai multe comenzi. Fiecare comandă include de obicei mai multe produse. Și fiecare comandă are cel puțin o plată - cel mai adesea o singură tranzacție cu cardul de credit, dar uneori clienții împart plățile.
**Eu:** ***Mulțumesc, Dana. Deci, dacă desenez asta ca model de nivel înalt, el ar arăta cam asta:***
**Eu:** ***Cu alte cuvinte, un Client plasează o Comandă, o Comandă conține unul sau mai multe Produse și o Comandă are cel puțin o Plată. Aceasta o include?***




**Dana:** Da, are sens. Așa funcționează procesul nostru de comandă la un nivel foarte înalt.
**Eu:** ***Mulțumesc, Dana. Asta e tot ce-mi trebuie deocamdată.***

**Diagrama de mai sus este un model conceptual foarte simplu**. Nu este sofisticat, și asta este intenționat. Pe baza interpretării tale a acestei conversații, ai putea desena totuși ceva diferit. Stai fără griji, totul e în regulă. Aici strălucește *Mecanismul de Dezvoltare Reală* în interacțiunile sale cu afacerea - îl poți arăta unui om de afaceri fără cunoștințe tehnice, de domeniul modelării, și acesta va înțelege ce descrii. De asemenea, nu există atribute, chei sau adnotări de cardinalitate (ce reflectă de exemplu numărul de relații 1:1,1:N, N:N, care pot fi incluse in ERD), deși unele abordări ale *Mecanismului de Dezvoltare Reală* le includ. În exemplul nostru, le vom adăuga în stratul logic. Este de remarcat faptul că, deși exemplul nostru nu conține informații de cardinalitate, acestea pot fi incluse în modelul conceptual dacă ajută la îmbunătățirea înțelegerii modelului de către tine și de către părțile interesate din afacerea ta. Deocamdată, avem un model de nivel înalt care surprinde conceptele fluxului de lucru al comenzilor pe care le-am discutat cu Dana.

Începând cu *Mecanismul de Dezvoltare Reală*, vom modela conceptele de business la nivel înalt: Clienți, Comenzi, Produse și Plată, într-un mod agnostic față de tehnologie. În acest moment, nu ne preocupă proiectarea tabelelor sau a câmpurilor. Pur și simplu captăm substantivele și verbele care ar putea fi esențiale pentru ceea ce părțile interesate consideră

important să ne spună. Când vă întâlniți cu părțile interesate, puteți desena acest lucru pe o singură foaie de hârtie, pe o tablă albă sau pe un instrument de diagramă. Pentru diagrama de mai sus, puteți schița aceasta și ca o diagramă de bază cu noduri: Client → Comandă → Produs și Plată. Implementarea este secundară captării informațiilor corecte și a modului în care lucrurile curg și se relaționează.

Folosind un limbaj obișnuit, substantivele sunt entitățile, iar verbele sunt relațiile. Să discutăm acest lucru puțin mai mult. În acest exemplu, avem următoarele entități: Client, Comandă, Produs și Plată. Să analizăm relațiile dintre aceste entități. Un Client *plasează* o Comandă, care *conține* Produse. O Comandă *are* asociată o Plată.

**Pentru claritate, iată un rezumat cu puncte.**

***Entități:***

- Client
- Comanda
- Produs
- Plată

***Relații:***

- Clientul plasează o comandă
- Comanda conține Produs(e)
- Comanda are plată(i)
- Clientul efectuează o plată

Aceasta ridică o întrebare interesantă pe care probabil o veți întâlni atunci când modelați date. În timpul interviului cu Dana, ea menționează cuvântul „client”. Ce este un client? Este cineva care a achiziționat vreodată un produs? Sau cineva care l-a achiziționat în ultimele 90 de zile? S-ar putea să fiți surprinși să aflați că un client poate însemna multe lucruri pentru mulți oameni. Aici se întâlnește modelarea datelor cu semantica. Aceeași etichetă de entitate poate reprezenta concepte diferite pentru departamente diferite, iar alinierea acestor definiții este la fel de importantă ca desenarea modelului. Cineva din operațiunile de comerț electronic va avea o idee despre client care diferă complet de cineva din marketing sau logistică. Este important să puneți întrebări ulterioare pentru a clarifica nuanțele, mai ales dacă auziți un termen folosit în mai multe unități de afaceri sau domenii. Comparați și contrastați ceea ce auziți cu alții din organizația dvs. (și, eventual, din afara acesteia). Obțineți cât mai multe informații posibil pentru a înțelege afacerea din mai multe perspective.

## Modelul logic (intermediar) – Designul intermediar

Avem o înțelegere conceptuală a procesului de comandă. Să aprofundăm în continuare și să construim modelul logic. Aici, veți dezvolta modelul conceptual și veți adăuga atribute, chei și cardinalități într-un mod agnostic față de tehnologie.

Pornind de la interviul simulat cu Dana, haideți să vorbim cu Mike, un stakeholder tehnic și inginer senior backend care gestionează baza de date RDBMS actuală pentru producție. Deși Mike nu a conceput în mod explicit un model conceptual, din fericire și-a amintit câteva aspecte ale normalizării bazelor de date din cursul său de baze de date de la universitate. Vom

acoperi normalizarea mai detaliat mai târziu în această carte, dar deocamdată, rețineți că normalizarea evită datele redundante, care pot crea inconsecvențe.

**Tu:** ***Mike, hai să coborâm cu un strat. Vreau să iau aceste concepte și să încep să adaug mai multă structură. Mă poți ajuta să înțeleg ce atribute are de obicei fiecare dintre aceste entități?***

**Mike:** Sigur. Tabelul clienți conține customer_id, first_name, last_name, email și created_at.

**Tu:** Deci, customer_id ar fi cheia primară aici?

**Mike:** Da, este un număr întreg. Toate cheile primare din baza de date sunt numere întregi.

**Tu:** Grozav. Pentru comenzi?

**Mike:** Tabelul Comenzi conține order_id, customer_id, order_date și status.

**Tu:** Și cum sunt stocate produsele? Le legați direct în comandă?

**Mike:** Nu, produsele în sine se află în tabelul Produse: product_id, name, sku, price și category. Există, de asemenea, un tabel OrderItems cu order_id și product_id, plus o cantitate asociată unei comenzi.

**Tu:** Perfect. Deci acesta este tabelul nostru de joncțiune pentru relația mai-mulți-la-mai-mulți dintre Comenzi și Produse?

**Mike:** Exact.

**Tu:** Dar plățile?

**Mike:** Acesta este un tabel separat de Plăți. Include payment_id, order_id, customer_id, amount, payment_method și payment_date.

**Tu:** Grozav. Voi schița un model logic în care sunt definite cheile primare și externe, fiecare entitate are atribute de bază și sunt afișate relațiile dintre entități. De asemenea, deocamdată îl voi păstra independent de platformă. Mulțumesc, Mike.

**Mike:** Sună bine. Mult succes.

Bine, deci avem câteva detalii despre ce se află în baza de date a aplicației. Să luăm entitățile din modelul nostru conceptual și să adăugăm atributele lui Mike. Vom adăuga și entitatea OrderItems.

**Iată entitățile și atributele lor**

- Client - id_client, prenume, nume, e-mail și created_at
- Comandă - id_comandă, id_client, data_comandă și stare
- Produs - ID_produs, nume, cod produs, preț și categorie
- ArticoleComandă - id_comandă și id_produs, cantitate
- Plată - id_plată, id_comandă, sumă, metodă_plată și dată_plată.

Să analizăm relațiile dintre entități și cheile lor. Cheile sunt o modalitate de a uni entitățile, așa cum am discutat anterior. Să schițăm cheile și modul în care entitățile se raportează între ele, folosind verbele din modelul conceptual ca ghid pentru relații.

**[Customer] (customer_id PK) --- (places) ---> [Order] (order_id, customer_id FK)**

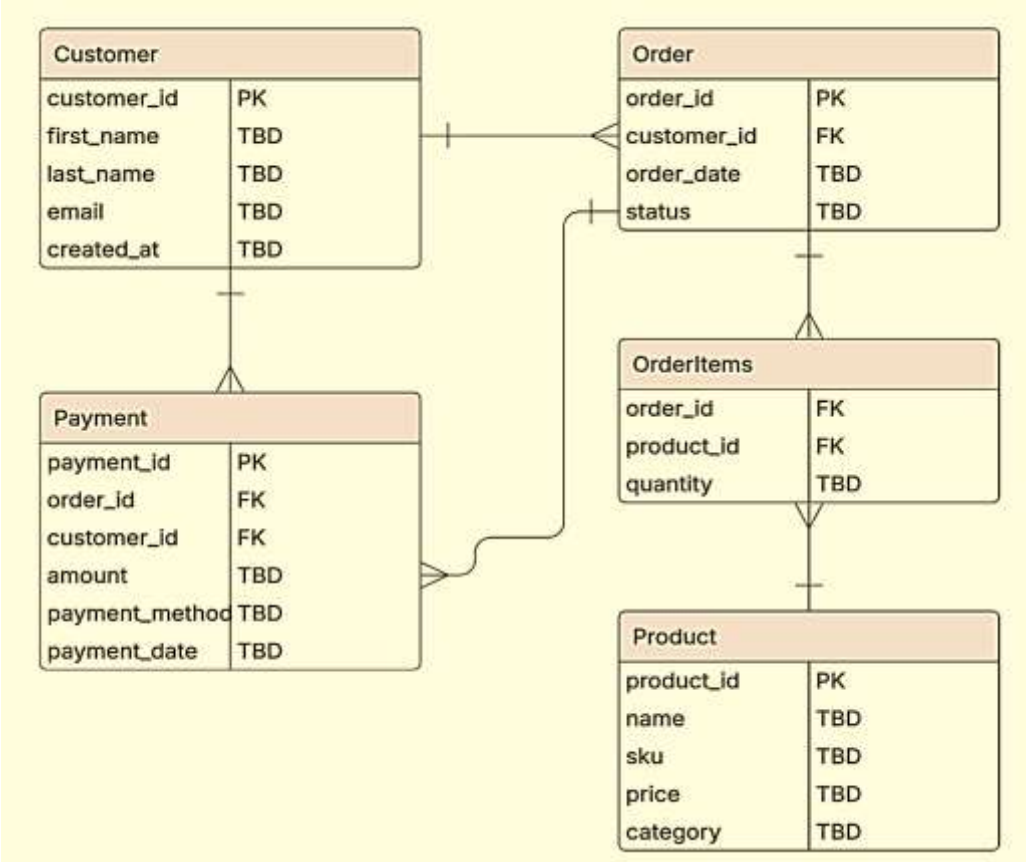
**[Order] (order_id PK) --- (contains) ---> [OrderItems] (order_id FK, product_id FK, Quantity)**

**[OrderItems] (order_id FK, product_id FK, Quantity) --- (contains) ---> [Product] (product_id PK)**

**[Order] (order_id PK) --- (has) ---> [Payment](order_id FK)**

**[Customer] (customer_id PK)  --- (makes) ---> [Payment] (customer_id FK)**

Să creăm acum un model logic care adaugă mai multe detalii CDM-ului, dar este totuși agnostic din punct de vedere tehnologic. Vom include entitățile și atributele acestora, cheile primare și externe, precum și cardinalitatea relațiilor (unu-la-mai-mulți, mai-mulți-la-mai-mulți etc.).




Acest model logic oferă un plan bun pentru implementare în orice RDBMS. Puteți extinde acest model la date semi-structurate (consultați capitolele noastre despre modelarea alt-relațională pentru idei).

Acum că am construit modelul logic, este timpul să dezvoltăm acest aspect la nivelul de modelare fizică.

## Modelul fizic de date

Ați capturat entitățile, numele atributelor, cheile și relațiile într-un model logic. Este timpul să aducem aceste idei în lumea reală cu o implementare fizică. Rețineți că lucrăm cu o bază de date implementată fizic. Haideți să vorbim din nou cu Mike, inginerul backend,

pentru a clarifica RDBMS-ul fizic, pentru a înțelege cum este proiectat și pentru a ne oferi informații despre crearea unei diagrame de model de date.
**Tu:** Acum că am stabilit modelul logic, haideți să discutăm despre implementarea sa concretă. Mike, poți să-mi explici deciziile tehnice cheie din baza de date actuală?
**Mike:** Sigur. Este o bază de date PostgreSQL. Folosim numere întregi pentru toate cheile primare. Câmpurile de e-mail sunt VARCHAR(255) și indexăm order_date în tabela Order deoarece aceasta este interogată frecvent.
**Tu:** Am înțeles. Sunt toate relațiile de cheie externă impuse în baza de date?
**Mike:** Da.
**Tu:** Bine de știut. Și există câmpuri opționale?
**Mike:** Da, payment_method poate fi modificat prin nulla deoarece înregistrăm comanda înainte de confirmarea plății.
**Tu:** OK. Deci, în modelul fizic, voi specifica lucruri de genul:

- Tipuri de date (VARCHAR(255), DECIMAL(10,2), etc.)
- Nulabilitate
- Indexuri
- Orice abateri de la modelul logic
- Note despre constrângeri sau comportamentul platformei

Sună corect? / Sună corect?
**Mike:** Exact. A, încă ceva. De asemenea, avem un timestamp created_at și un timestamp updated_at pentru fiecare tabel. Cu excepția OrderItems.
**Tu:** În discuția noastră anterioară, nu ai menționat că fiecare tabel are created_at și update_at. Dar bună idee. Voi nota asta în documentația schemei.
**Mike:** Scuze, am uitat. Dar da, câmpurile created_at și update_at sunt în ambele tabele.

**Tu:** Nicio problemă, Mike. Mulțumesc pentru clarificare. Voi nota. Și mulțumesc pentru timpul acordat. Acest lucru a fost foarte util.

**Mike:** Sună bine. Cât timp ești aici, ar trebui să-ți dau acces de citire la GitHub, unde păstrăm schema bazei de date PostgreSQL?
**Tu:** Uau, ar fi uimitor. Mulțumesc, Mike.
**Mike:** Excelent. Sună dacă mai ai și alte întrebări.
Discuția ta cu Mike a oferit niște informații utile. În notițele întâlnirii, menționezi baza de date (PostgreSQL) și necesitatea completării tipurilor de date, indexurilor, câmpurilor care acceptă valori NULL și constrângerilor. Și mai bine, acum ai acces la limbajul de definire a datelor (DDL) SQL. Bravo!

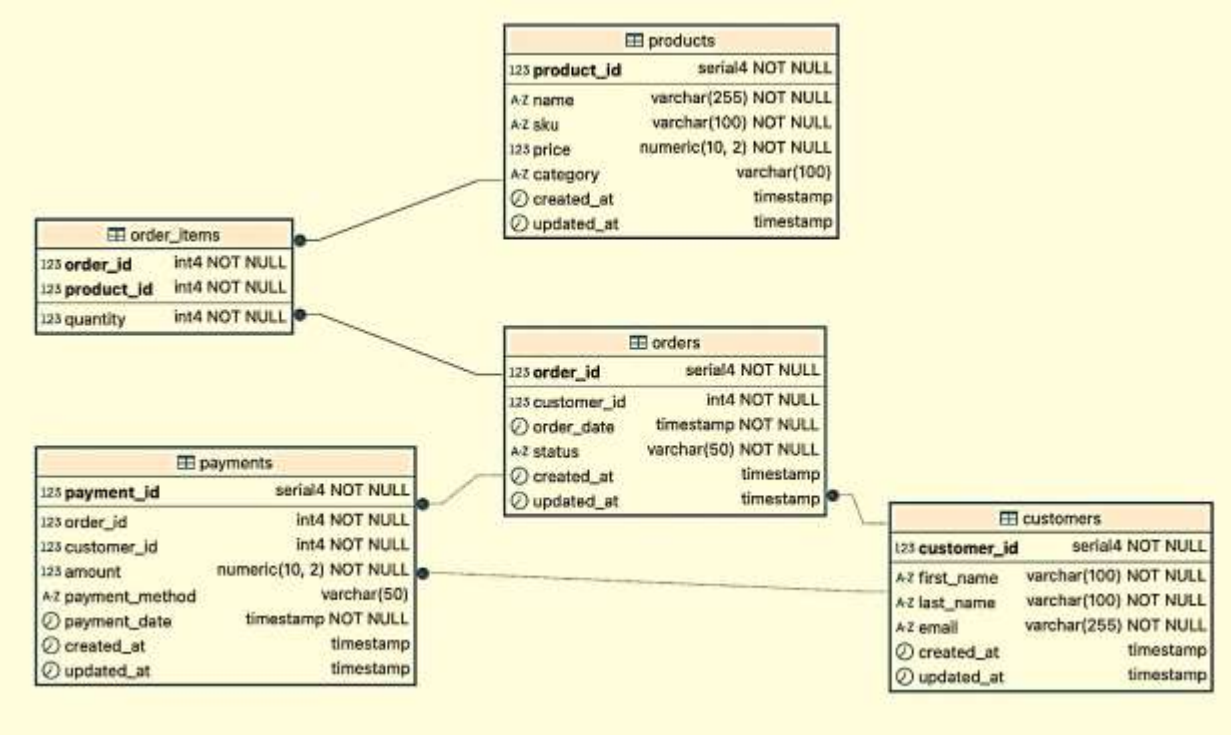
**Hai să aruncăm o privire la DDL-ul pe care ți l-a furnizat Mike.**

```
-- Customers Table
CREATE TABLE customers (
    customer_id SERIAL PRIMARY KEY,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
```

```
    email VARCHAR(255) UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Orders Table
CREATE TABLE orders (
    order_id SERIAL PRIMARY KEY,
    customer_id INTEGER NOT NULL REFERENCES
customers(customer_id),
    order_date TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(50) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Products Table
CREATE TABLE products (
    product_id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    sku VARCHAR(100) UNIQUE NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    category VARCHAR(100),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- OrderItems Table (Junction Table)
CREATE TABLE order_items (
    order_id INTEGER NOT NULL REFERENCES orders(order_id),
    product_id INTEGER NOT NULL REFERENCES
products(product_id),
    quantity INTEGER NOT NULL CHECK (quantity > 0),
    PRIMARY KEY (order_id, product_id)
);

-- Payments Table
CREATE TABLE payments (
    payment_id SERIAL PRIMARY KEY,
    order_id INTEGER NOT NULL REFERENCES orders(order_id),
    customer_id INTEGER NOT NULL REFERENCES
customers(customer_id),
    amount DECIMAL(10, 2) NOT NULL,
    payment_method VARCHAR(50),  -- Nullable until confirmed
```

```
    payment_date TIMESTAMP NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Indexes for performance
CREATE INDEX idx_orders_order_date ON orders(order_date);
CREATE INDEX idx_payments_order_id ON payments(order_id);
```

Iată diagrama fizică a bazei de date generată de un editor PostgreSQL.




Din nou, vom crea diagrama DDL și a modelului fizic într-o situație greenfield. Din fericire, Mike avea DDL-ul la îndemână, iar modelul de date arată decent.

V-ați putea întreba dacă ar fi să începeți cu diagrama DDL și a bazei de date fizice și să lucrați înapoi până la modelul conceptual. Da, puteți. Modelul fizic de date nu a apărut pur și simplu ca prin magie. A fost creat de cineva sau de un grup de persoane care, sperând, încercau să mapeze concepte de afaceri la un model de date. Uneori există un model de date intenționat, dar adesea diagrama fizică ar putea fi tot ce aveți. Dacă acesta este cazul, utilizați această diagramă și lucrați înapoi, făcând inginerie inversă înapoi la modelul conceptual. În toate cazurile, consultați părțile interesate din domeniu și cele tehnice pentru a răspunde la întrebări și a clarifica înțelegerea comună a ceea ce reprezintă modelul de date.

Acum aveți o idee de bază despre cum să aplicați nivelurile de modelare a datelor în sensul tradițional. Deoarece acești pași sunt bine documentați în nenumărate cărți și articole

despre proiectarea bazelor de date, v-am oferit o prezentare generală la nivel general, deoarece acest subiect a fost tratat în detaliu în nenumărate locuri. Pentru a înțelege mai multe, vă rugăm să consultați secțiunea Lecturi suplimentare de la sfârșitul acestui capitol.

Deși aplicarea nivelurilor de modelare a datelor la un caz de utilizare tradițional, cum ar fi proiectarea unei baze de date relaționale (sau a unui depozit de date), reprezintă o bază bună, lumea datelor constă în mare parte din date nestructurate. În secțiunea următoare, vom analiza cum se aplică nivelurile la datele nestructurate.