

Databázové systémy

Database design – návrh databázovej štruktúry

Motivácia: Prečo na správnom návrhu záleží?

- Predstavte si e-shop, kde pri zmene priezviska zákazníčky musíte ručne prepísať 50 riadkov s jej starými objednávkami, alebo kde výpadok prúdu uprostred platby rozbije celé účtovníctvo.
- **Cieľ prednášky:** Navrhnuť databázovú štruktúru nášho e-shopu tak, aby zvládlo aj Black Friday a nezrútilo sa pri prvej objednávke.

Motivácia: Prečo na správnom návrhu záleží?

- **Databáza ako základy domu:** zlý základ zrúti celú stavbu.
- **Dôsledky zlého návrhu:** pomalé dopyty, duplicita dát, anomálie pri vkladaní a mazaní.
- **Cieľ dizajnéra:** efektivita, integrita a škálovateľnosť.

Dátový model

- Je to **popis štruktúry dát**, ich **typov**, **ohraničení** a **vzťahov** poskytnutím definície a formátu dát
- Poskytuje nutné nástroje na dosiahnutie dátovej **abstrakcie** – ako sa symboly a koncepty vzťahujú k reálnemu svetu, skrývajú štruktúru a parametre fyzickej pamäte
- Väčšina dátových modelov obsahuje aj množinu základných **operácií** pre špecifikáciu dopytov a aktualizáciu databázy

Tri typy dátových modelov

- **Konceptuálny (Čo ideme ukladať? Biznis logika, nezávislá od softvéru):** Chceme, aby si *Zákazník* mohol kúpiť *Produkt* cez *Objednávku*.
- **Logický (Ako to prepojíme? Tabuľky, kľúče, mapovanie vzťahov):** Prepojíme tabuľku *zákazníkov* s *objednávkami* pomocou kľúčov.
- **Fyzický (Kde a ako to reálne uložíme? Konkrétny DBMS, indexy, dátové typy):** Použijeme MySQL, stĺpcu *cena* dáme typ DECIMAL(10,2) a na *kod_produkta* zavesíme index pre rýchle vyhľadávanie.

Konceptuálny dátový model

- **Abstraktný** pohľad na dáta nezávislý od hardvéru či konkrétneho databázového softvéru (DBMS).
- Mapuje čistú biznis logiku.
- Je to séria tvrdení o povahe databázy (napr. organizácie), ktorých význam je opísaný **názvami entít** a ich **vzťahmi**, ktoré sú založené na požiadavkách a analýze.
- Mali by byť nezávislé (ERD = Entity-Relationship Diagram, UML = Unified Modeling Language) od konkrétnej technológie manažmentu dát.

Entita

- **Entitný typ/Entita:** Množina objektov reálneho sveta s rovnakými vlastnosťami (atribútmi).
 - Napr. **ZÁKAZNÍK, PRODUKT**.
 - V ER diagrame sa znázorňuje **obdĺžnikom**.
- **Atribút:** Konkrétna vlastnosť alebo charakteristika entitného typu.
 - Napr: **meno, cena**.
 - V ER diagrame sa znázorňuje **oválom/elipsou**.
- **Inštancia entity:** Jeden konkrétny, jednoznačne identifikovateľný výskyt entitného typu.
 - Napr. Produkt **iPhone 16 s ID 987**.
- Namiesto dvojice **entita** a **inštancia entity** sa niekedy používa dvojica **entitný typ** a **entita**.



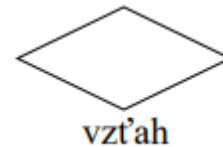
entita



atribút

Vzťah

- **Vzťah (Relationship):** Logická väzba, asociácia alebo interakcia medzi dvoma alebo viacerými **entitnými typmi**. Vyjadruje, ako objekty z reálneho sveta navzájom korelujú alebo na seba vplývajú.
- **Vzťahový typ (Relationship Type):** Abstraktný predpis alebo štruktúra prepojenia
 - Napr. ZÁKAZNÍK **vytvára** OBJEDNÁVKU.
 - V ER diagrame sa znázorňuje **kosoštvorcom**.
- **Inštancia vzťahu (Relationship Instance):** Konkrétne prepojenie dvoch špecifických inštancií entít
 - Napr. Zákazník *Ján Mrkvička* vytvoril objednávku č. 2026001.



Charakteristika vzťahu

- Pri modelovaní relačných databáz dôležitú úlohu zohráva početnostná charakterizácia vzťahu, ktorá vyjadruje, že jednému riadku tabuľky koľko partnerských riadkov inej tabuľky zodpovedá.
- V ER diagramoch početnosť môžeme popísať **kardinalitou** a **účasťou**.
- **Kardinalita** vyjadruje **maximálny** počet partnerských inštancií a **účasť** **minimálny**.
- **Kardinalita** sa zadáva hodnotami **1, N** a **účasť** hodnotami **0, 1**

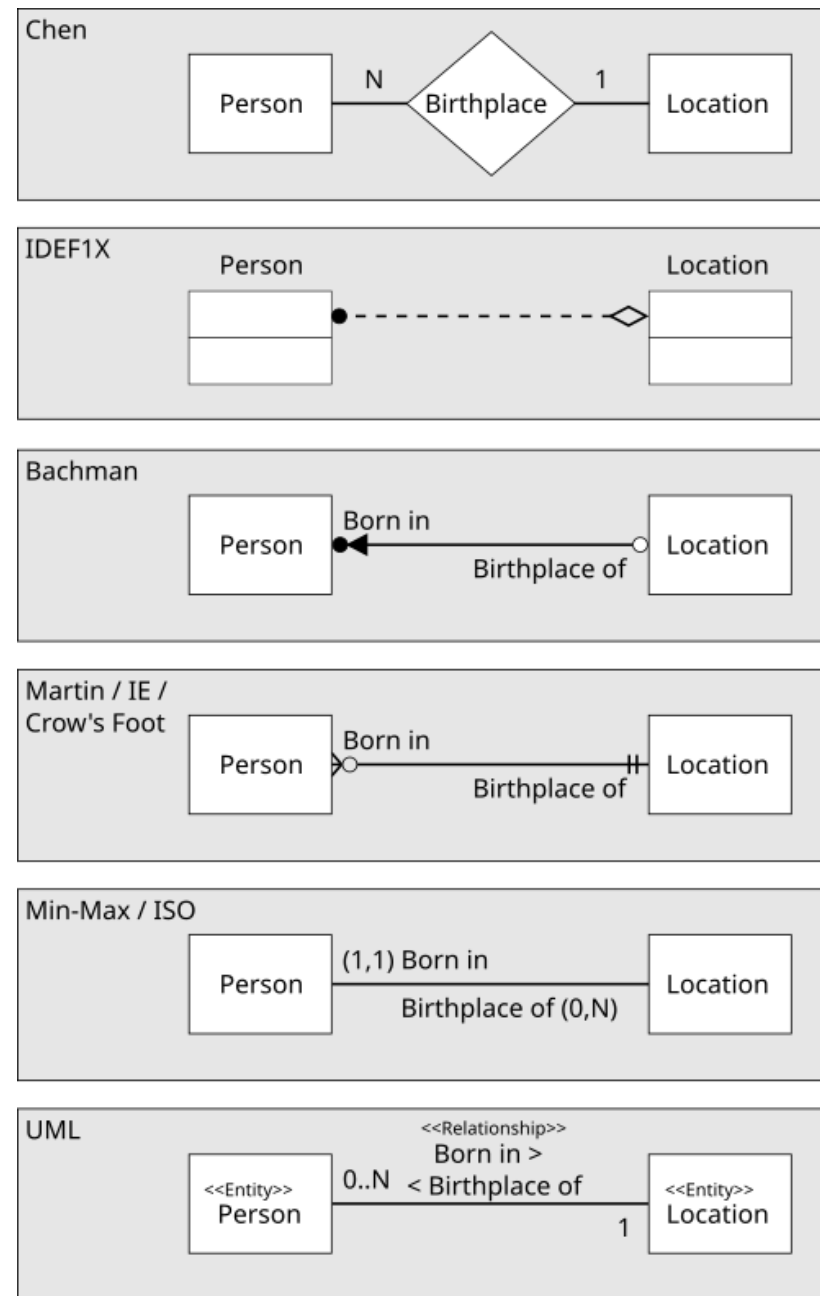
1. Kardinalita (Cardinality)

- Určuje **maximálny počet** inštancií jedného entitného typu, ktoré sa môžu podieľať na vzťahu s jednou inštanciou druhého entitného typu.
 - **1:1 (Jeden k jednému):** Inštancia A sa viaže najviac na jednu inštanciu B a naopak.
 - Napr. *Vedúci/Manažér* vedie *Pobočku e-shopu*.
 - **1:N (Jeden k mnohým):** Inštancia A sa môže viazať na viacero inštancií B, ale inštancia B iba na jednu inštanciu A.
 - Napr. *Zákazník* vytvára *Objednávky*.
 - **M:N (Mnohí k mnohým):** Inštancia A sa viaže na viacero inštancií B a naopak
 - Napr. *Objednávka* obsahuje *Produkty*.

2. Parcialita/Účasť (Participation/Optionality)

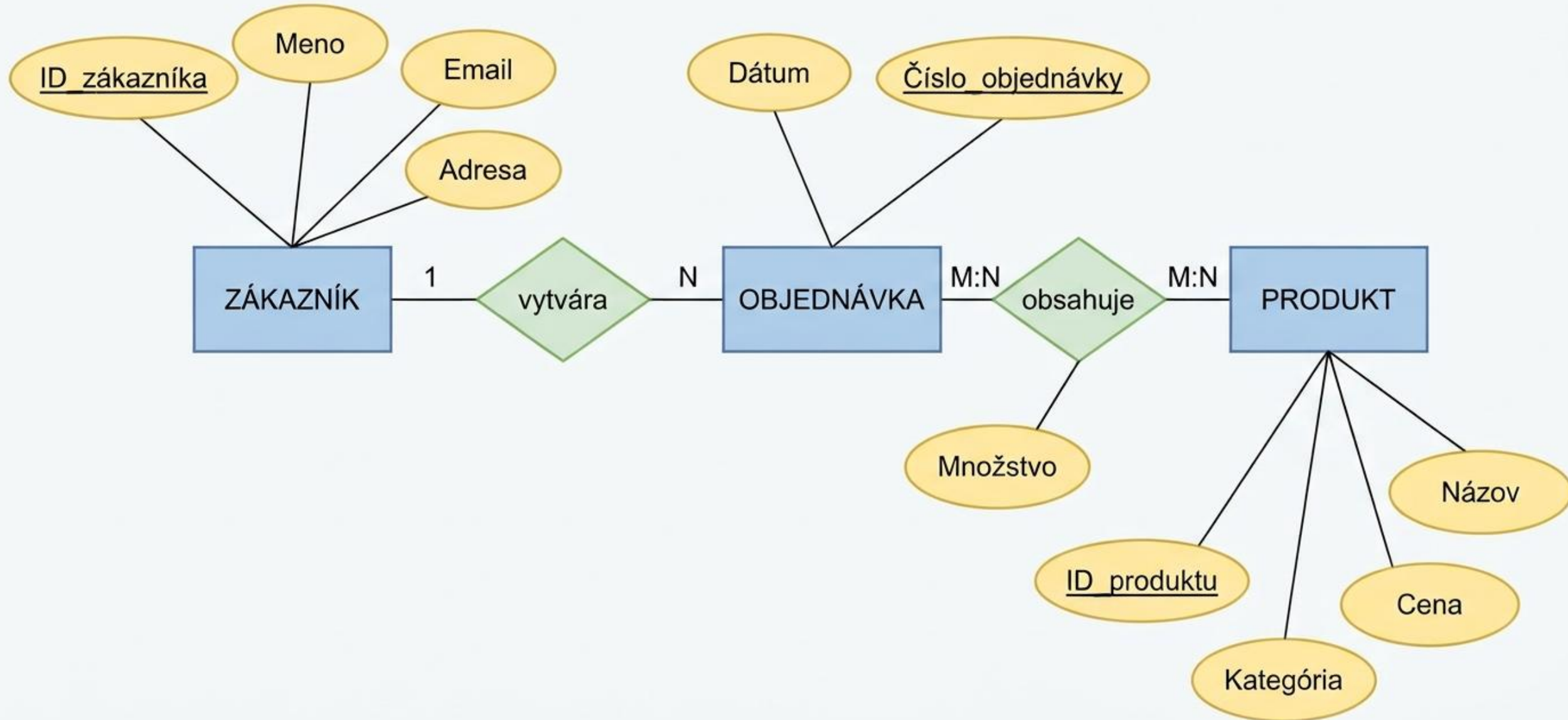
- Určuje **minimálny počet** inštancií, ktoré sa musia na vzťahu zúčastniť (či je vzťah povinný alebo voliteľný).
 - **Total (Povinná účasť) - 1**: Každá inštancia entity sa musí podieľať na vzťahu.
 - Napr. Každá objednávka **musí** patriť nejakému zákazníkovi (v relačnom modeli to znamená NOT NULL cudzí kľúč).
 - **Partial (Voliteľná účasť) - 0**: Môžeme mať niekoľko inštancií entít, ktoré sa nepodieľajú na vzťahu.
 - Napr. Nový produkt v katalógu ešte **nemusí** byť súčasťou žiadnej objednávky.

Značenie vzťahov



https://thumb.wikimedia.org/wikipedia/commons/thumb/f/f1/ERD_Representation.svg/500px-ERD_Representation.svg.png

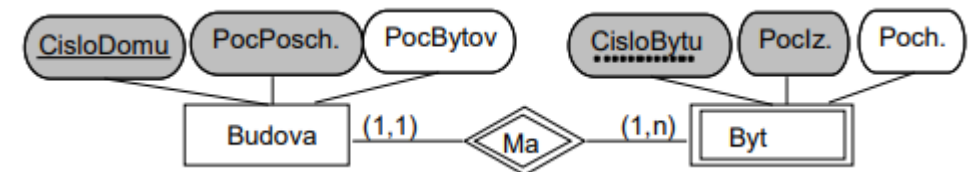
Konceptuálny dátový model E-shopu (Chenova notácia)



Silná a slabá entita

- **Silná entita (Nezávislá):** Dokáže existovať sama o sebe a má svoj **vlastný kľúč**, ktorý ju jednoznačne identifikuje (napr. ZÁKAZNÍK s jedinečným emailom alebo PRODUKT s ID produktu).
- **Slabá entita (Závislá):** **Nedokáže existovať bez existencie inej (silnej) entity** a nemá dostatok vlastných atribútov na to, aby vytvorila samostatný **primárny kľúč**. Je existenčne aj identifikačne **závislá od svojho vlastníka**.
- **Notácia v ERD:** Silná entita sa kreslí ako klasický obdĺžnik, slabá entita ako **dvojitý obdĺžnik** a vzťah k nej ako **dvojitý kosoštvorec**.

- Je objednávka silná alebo slabá entita?
Prečo áno/nie?



Logický dátový model

- Transformácia konceptuálneho modelu do konkrétnej dátovej štruktúry (napr. relačné tabuľky), stále však nezávislá na konkrétnom DBMS.
- Detailný popis dátovej štruktúry vyjadrený formou databázových entít (**tabuliek**), ich atribútov (**stĺpcov**) a logických **vzťahov** medzi nimi. Je plne nezávislý od konkrétneho databázového softvéru (DBMS), ale rešpektuje pravidlá relačného modelu (napr. **neexistujú** tu vzťahy **M:N**).
- Obohacuje konceptuálny dátový model o informácie o **doméne** a **dátových štruktúrach**. Zahŕňa **atribúty pre všetky entity**, **primárne a cudzie kľúče**. Na tomto stupni sa robí **normalizácia**.

Logický dátový model

- **Entity sa menia na tabuľky:** Už pracujeme s presnou štruktúrou riadkov a stĺpcov.
- **Primárne kľúče (PK):** Každá tabuľka musí mať jasne označený stĺpec (alebo skupinu stĺpcov), ktorý garantuje unikátnosť riadkov.
- **Cudzie kľúče (FK):** Vzťahy sa už nekreslia ako kosoštvorce, ale realizujú sa pomocou stĺpcov cudzích kľúčov, ktoré fyzicky prepájajú tabuľky.
- **Rozbité vzťahy M:N:** Spojovacie tabuľky sú už priamo zapísané v modeli.

Logický dátový model

- **Primárny kľúč (Primary Key - PK):** Atribút alebo množina atribútov, ktorá jednoznačne identifikuje každú inštanciu entity (riadok) v tabuľke. Na úrovni databázy preň platia dve nekompromisné pravidlá:
 - **Unikátnosť (UNIQUE):** Žiadne dva riadky nesmú mať rovnakú hodnotu PK.
 - **Nezápornosť/Povinnosť (NOT NULL):** Hodnota primárneho kľúča nesmie nikdy ostať prázdna.
- **Cudzí kľúč (Foreign Key - FK):** Atribút alebo množina atribútov v jednej tabuľke (dcérskej), ktorého hodnoty musia korešpondovať s hodnotami primárneho kľúča (PK) v inej tabuľke (materskej), alebo musia byť prázdne (NULL), ak je vzťah voliteľný.
 - Napr. Máme materskú tabuľku **ZAKAZNIK** (kde je PK **zakaznik_id**) a dcérsku tabuľku **OBJEDNAVKA**. Aby sme vedeli, kto objednávku vytvoril, do tabuľky **OBJEDNAVKA** pridáme stĺpec **zakaznik_id** ako **cudzí kľúč (FK)**.
- **Doména (Domain):** Kombinácia typu a ohraničenia, ktoré definuje množinu možných hodnôt atribútu.
 - Napr. pohlavie **CHAR(1) NOT NULL CHECK (pohlavie IN ('m', 'z'))**

Integrita dát

- **Integritné obmedzenia:** Súbor pravidiel a podmienok, ktoré garantujú, že dáta v databáze sú **korektné** v súlade s definovanými obmedzeniami **bez nedostatku** a **logicky konzistentné bez protirečení**. Databáza by mala obsahovať iba **platné hodnoty** (prípustné, správne – valid, accurate, correct).

Integrita dát

Typ integrity	Čo definuje?	Príklad z E-shopu
Integrita entít	Každá tabuľka musí mať unikátny Primárny kľúč (PK), ktorý nesmie byť NULL.	Nemôžu existovať dve objednávky s rovnakým id_objednavky.
Referenčná integrita	Cudzí kľúč (FK) musí odkazovať na existujúci PK alebo byť prázdny.	Systém nedovolí priradiť objednávku neexistujúcemu zákazníkovi.
Doménová integrita	Hodnoty v stĺpci musia patriť do množiny povolených hodnôt.	V stĺpci množstvo nemôže byť záporné číslo alebo text.

Spojovacia (asociačná) tabuľka

- V konceptuálnom modeli (ERD) je vzťah OBJEDNÁVKA —(obsahuje)— PRODUKT v poriadku.
- Relačné databázy (SQL) však dokážu pracovať len so vzťahmi 1:1 a 1:N.
- Každý vzťah typu **M:N** musíme v **logickom dátovom modeli** rozbiť na **dva vzťahy typu 1:N**. Na to využijeme nový entitný typ, ktorý vložíme do stredu.
- **Spojovacia (Asociačná) tabuľka:** Špeciálna tabuľka, ktorej primárnym cieľom je prepojiť dve iné tabuľky vo vzťahu M:N. Obsahuje minimálne dva cudzie kľúče (FK), ktoré odkazujú na primárne kľúče (PK) oboch prepájaných tabuliek.

Konceptuálny dátový model (ERD)



Logický dátový model (Dekompozícia)



Spojovacia (asociačná) tabuľka

- Je objednávka silná alebo slabá entita?
Prečo áno/nie?
- Je položka_objednávky silná alebo slabá entita?
Prečo áno/nie?

Normalizácia dát

- **Normalizácia** je proces eliminovania duplicít, nekonzistentností v dátach a závislostiach v relačných databázach pomocou
 - **pravidiel medzi atribútmi a funkčnými závislosťami**
 - **dekompozície** tabuľky (náhrada tabuľky s veľkým počtom stĺpcov s viacerými tabuľkami s menším počtom stĺpcov)s cieľom zabezpečiť integritu dát a úsporne šetriť miesto na ukladanie údajov.

- **Ciele:**
 - Eliminácia redundancie dát (duplicitného ukladania)
 - Minimalizácia rizika vzniku anomálií (Insert, Update, Delete)
 - Zabezpečenie logickej konzistentnosti dát
- **Pravidlo:** Uložiť každú jednu informáciu do databázy práve raz.

Normalizácia dát

- Normalizácia prebieha v po sebe idúcich krokoch, ktoré nazývame **Normálne formy (NF)**. Aby bola databáza považovaná za zdravú, musí v komerčnej praxi spĺňať prvé tri:
- **1. Normálna forma (1NF):** Všetky atribúty musia byť atomárne (nedeliteľné). V bunke nesmie byť zoznam hodnôt (napr. v stĺpci meno nemôže byť napísané 'Adam a Eva'). Každý riadok musí byť unikátny (musí existovať primárny kľúč).
- **2. Normálna forma (2NF):** Splnená 1NF + všetky neklúčové stĺpce musia byť plne závislé od celého primárneho kľúča (rieši sa najmä pri zložených kľúčoch).
- **3. Normálna forma (3NF):** Splnená 2NF + medzi neklúčovými stĺpcami nesmie existovať tranzitívna závislosť. To znamená, že žiadny neklúčový stĺpec nesmie závisieť od iného neklúčového stĺpca (napríklad, ak by ste mali v tabuľke stĺpce PSC a Mesto, porušujete 3NF, pretože názov mesta viete odvodiť z PSC a tieto dáta patria do samostatnej číselníkovej tabuľky).
- **Heat-Boyce-Coddova normálna forma (HBCNF):** Sprísňuje tretiu normálnu formu tým, že vyžaduje, aby každý stĺpec, ktorý určuje hodnotu iného stĺpca, bol striktne kandidátskym kľúčom.

Normálne formy

- **4. Normálna forma (4NF):** Eliminuje takzvané viachodnotové závislosti, čím zabraňuje tomu, aby sa v jednej tabuľke museli zbytočne kombinovať dva nezávislé vzťahy typu 1:N pre tú istú entitu.
- **Essential Tuple Normálna Forma (ETNF):** Redukuje nadbytočnosť dát tým, že striktne oddeľuje funkčné závislosti od závislostí spojenia, čím zabezpečuje, že relácia nemôže byť dekomponovaná bez straty dôležitých štrukturálnych väzieb.
- **5. Normálna forma (5NF):** Zabezpečuje, že zložité cyklické vzťahy medzi viacerými entitami je možné bezstratovo rozdeliť do samostatných tabuliek a následne ich bez vzniku falošných riadkov spojiť pomocou operácie JOIN.
- **Doménovo-klúčová normálna forma (DKNF):** Predstavuje teoretické maximum normalizácie, kedy sú všetky biznis pravidlá a integritné obmedzenia priamym dôsledkom definície kľúčov a povolených hodnôt stĺpcov, čo absolútne vylučuje akékoľvek anomálie.
- **6. Normálna forma (6NF):** Predstavuje dekompozíciu dotiahnutú do absolútneho extrému, kedy každá tabuľka obsahuje iba primárny kľúč a maximálne jeden ďalší neklúčový stĺpec, čo nachádza uplatnenie najmä v temporálnych databázach sledujúcich históriu zmien v čase.

Normálne formy

- Normalizácia je inherentne **inkluzívny a kumulatívny proces**.
Relácia **nemôže** dosiahnuť vyššiu normálnu formu (napr. 4NF) bez toho, aby predtým stopercentne spĺňala podmienky **všetkých predchádzajúcich úrovní** (1NF → 2NF → 3NF → HBCNF → 4NF...).
- Vyššie normálne formy stavajú na základoch tých nižších a ďalej ich sprísňujú.

Normálna forma	Hlavný cieľ (Čo odstraňuje / Čo vynucuje)	Typická anomália / Riešený problém
1NF	Vynucuje atomicitu hodnôt atribútov a existenciu primárneho kľúča.	Odstraňuje zoznamy a duplicitné riadky.
2NF	Odstraňuje čiastočnú závislosť neklúčových stĺpcov na zloženom kľúči.	Rieši redundantné opakovanie dát pri zložených kľúčoch.
3NF	Odstraňuje tranzitívnu závislosť medzi neklúčovými stĺpcami.	Zamedzuje, aby stĺpec A určoval hodnotu stĺpca B mimo kľúča.
HBCNF	Zaisťuje, že každý determinant (určujúci stĺpec) musí byť superkľúčom.	Rieši prekryvajúce sa zložené kandidátske kľúče.
4NF	Odstraňuje viachodnotové závislosti (MVD).	Zamedzuje nutnosti držať v tabuľke kombinácie dvoch nezávislých vzťahov 1:N.
5NF	Odstraňuje anomálie pri viacnásobnom cyklickom spájaní relácií.	Zaručuje 100 % bezstratovú dekompozíciu a opätovný JOIN.
DKNF	Zjednocuje všetky pravidlá pod definície domén a kľúčov.	Absolútna eliminácia akýchkoľvek logických anomálií modifikácie.

UNF (Unnormalized form)

- Nenormalizovaná forma

- **Problém:** V bunke produkty máme zoznam hodnôt oddelených čiarkami (dáta nie sú atomické).

id_objednavky	zakaznik_email	mesto_dorucenia	produkty
101	jan@email.sk	Košice	iPhone 16 (1ks), Obal (2ks)
102	anna@email.sk	Prešov	Obal (1ks)

1NF (Prvá normálna forma)

- Prechod od neatomických hodnôt (UNF)

- **Definícia:** Tabuľka spĺňa 1NF a **všetky neklúčové atribúty sú plne funkčne závislé od celého primárneho kľúča**, nie iba od jeho časti. (Rieši sa iba vtedy, ak má tabuľka zložený primárny kľúč).
- **UNF:** id_objednavky (101) → produkty: 'iPhone 16 (1ks), Obal (2ks)'
- **Riešenie:**
 - Každý produkt v objednávke má svoj vlastný samostatný riadok.
 - Zavedenie zloženého /kompozitného PK: (id_objednavky, id_produkť)
- **Pretrvávajúci problém:** Čiastočné funkčné závislosti.
 - Cena produktu závisí len od názvu produktu, email zákazníka len od id_objednavky.

2NF (Druhá normálna forma)

- Odstránenie čiastočných závislostí

- **Definícia:** Tabuľka spĺňa 1NF a **všetky neklúčové atribúty sú plne funkčne závislé od celého primárneho kľúča**, nie iba od jeho časti. (Rieši sa iba vtedy, ak má tabuľka zložený primárny kľúč).
- **Riešenie:**
 - Identifikujeme **čiastočné závislosti** (keď stĺpec závisí iba od jednej zložky zloženého kľúča). Tieto stĺpce vyčleníme do nových, samostatných tabuliek spolu s tou časťou kľúča, od ktorej závisia.
- **Výsledok:**
 - Tabuľka PRODUKTY: produkt (PK), cena_produkту
 - Tabuľka OBJEDNÁVKY_A_ZÁKAZNÍCI: id_objednavky (PK), zakaznik_email, mesto_dorucenia
- **Pretrvávajúci problém:** Tranzitívna závislosť v druhej tabuľke (mesto závisí od emailu zákazníka, nie priamo od objednávky).

3NF (Tretia normálna forma)

- Odstránenie tranzitívnych závislostí

- **Definícia:** Tabuľka spĺňa 2NF a **žiadny neklúčový atribút nie je tranzitívne závislý od primárneho kľúča**. To znamená, že neklúčové stĺpce nesmú závisieť od iných neklúčových stĺpcov.
- **Riešenie:**
 - Nájdeme závislosti medzi neklúčovými stĺpcami. Vyčleníme ich do novej číselníkovej tabuľky. Stĺpec, ktorý bol „v strede“ závislosti, ponecháme v pôvodnej tabuľke ako cudzí kľúč (FK).
- **Výsledok:**
 - Tabuľka ZÁKAZNÍCI: zakaznik_email (PK), mesto_dorucenia
 - Tabuľka OBJEDNÁVKY: id_objednavky (PK), zakaznik_email (FK), datum

HBCNF (Heat-Boyce-Coddova normálna forma)

- Prísna verzia 3NF pre prekrývajúce sa kľúče

- **Definícia:** Tabuľka spĺňa 3NF a pre každú netriviálnu funkčnú závislosť typu $X \rightarrow Y$ platí, že **X musí byť nadkľúčom** danej tabuľky. HBCNF je o niečo prísnejšia verzia 3NF a rieši anomálie, ktoré môžu nastať, ak tabuľka obsahuje **viacero zložených kandidátnych kľúčov, ktoré sa navzájom prekrývajú**.
- **Problémový príklad (3NF, ale nie BCNF):** Máme tabuľku SKLADOVÉ_ZÁSOBY, v ktorej evidujeme, ktorý dodávateľ dodáva aký typ tovaru do konkrétneho skladu. V e-shope platia tieto **dve prísne biznis pravidlá**:
 - Pre každý sklad a pre každý typ produktu existuje **iba jeden výhradný dodávateľ** (napr. elektroniku do skladu v Košiciach vozí výhradne Sony).
 - Každý dodávateľ zásobuje **iba jeden jediný konkrétny sklad** (napr. Sony má zmluvu len so skladom v Košiciach, do Bratislavy nevozí nič).
- Pozrime sa na funkčné závislosti (kto čo jednoznačne určuje):
 - (mesto_skladu, typ_produkту) $\rightarrow$ dodavateľ (V poriadku, ľavá strana je primárny kľúč).
 - dodavateľ $\rightarrow$ mesto_skladu (Keďže Sony zásobuje *iba* Košice, samotný dodávateľ jednoznačne určuje mesto skladu).
- Na ľavej strane druhej závislosti stojí dodavateľ. Avšak stĺpec dodavateľ **nie je nadkľúčom** tejto tabuľky (neidentifikuje celý riadok samostatne). Podľa definície BCNF musí byť *každá* ľavá strana funkčnej závislosti nadkľúčom.
- **Anomália mazania:** Predstavte si, že e-shop úplne vypredá kategóriu *Knihy* zo skladu v Košiciach a rozhodne sa tento riadok z databázy vymazať. Spolu s vymazaním kategórie kníh sme z databázy kompletne vymazali informáciu o tom, že *Martinus je naším partnerským dodávateľom pre sklad v Košiciach*. Ak budeme chcieť o mesiac naskladniť do Košíc učebnice, e-shop stratil informáciu, akého dodávateľa má kontaktovať.
- **Riešenie:**
 - Ak zistíme, že neklúčový stĺpec (alebo časť zloženého kľúča) určuje hodnotu iného kľúčového stĺpca (determinant nie je kandidátny kľúč), musíme tabuľku opäť rozdeliť na dve.
- **Výsledok:**
 - Tabuľka DODÁVATELIA_A_SKLADY: dodavateľ (PK), mesto_skladu
 - Tabuľka DODÁVKY_PRODUKTOV: dodavateľ (PK1/FK), typ_produkту (PK2)

NF

• ...

Fyzický dátový model

- Finálna úroveň modelovania dát, ktorá popisuje **reálnu implementáciu databázovej štruktúry v konkrétnom systéme riadenia bázy dát** (DBMS, napr. PostgreSQL, MySQL). Obsahuje presné priradenie dátových typov, fyzické kľúče, unikátne indexy, spúšťače (triggers) a obmedzenia (constraints).
- Obsahuje všetky tabuľkové štruktúry, vrátane názvov stĺpcov, ich dátových typov a ohraničení. Môžeme ho považovať za mapovanie logického dátového modelu na databázový model daného DBMS (Database Management System).

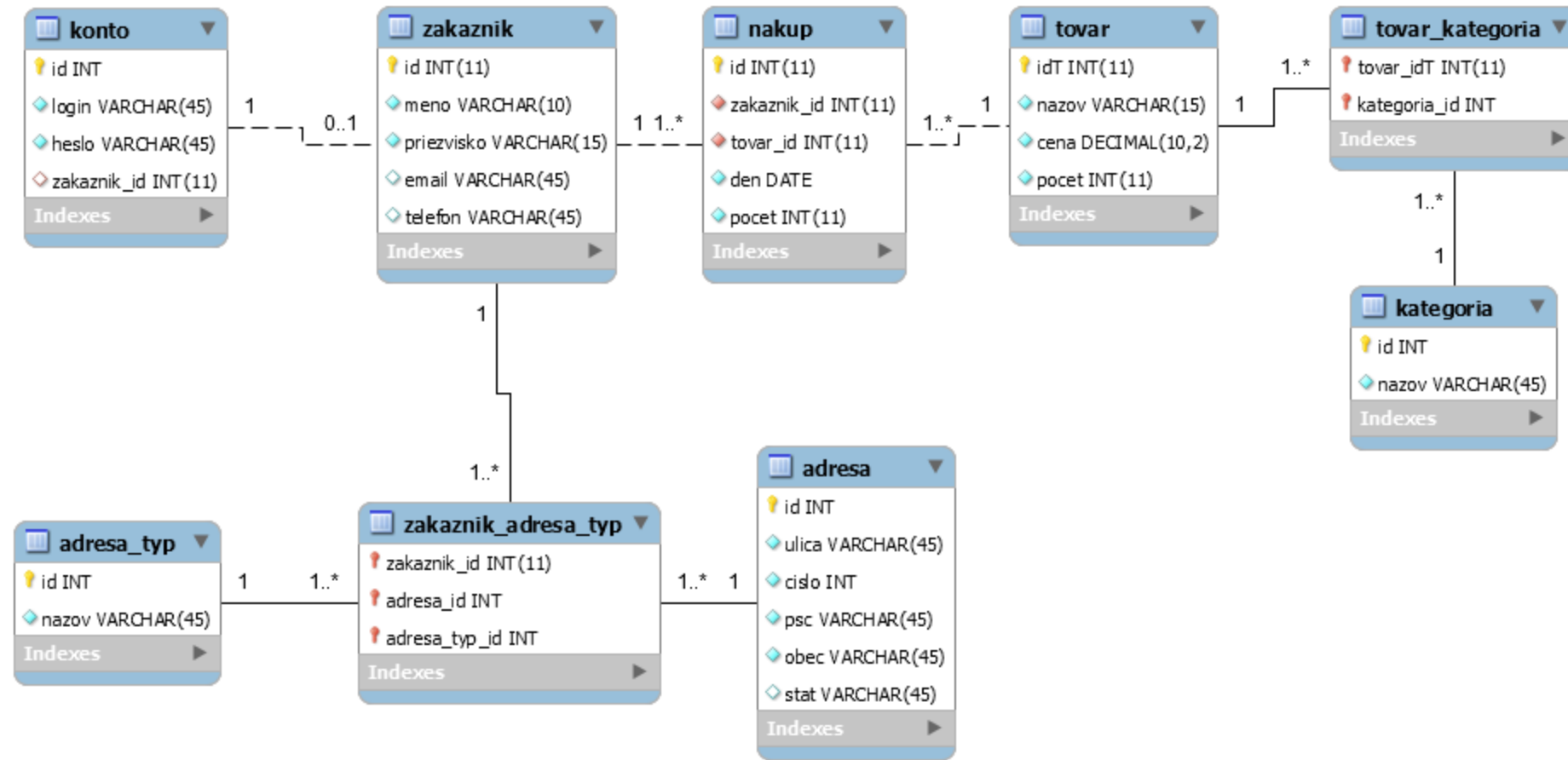
Čo sa zmenilo oproti logickému modelu?

- **Atribúty majú dátové typy:** Riešime efektivitu ukladania na disk (INT, VARCHAR, TIMESTAMP).
- **Technické vynútenie integrity:** Logická integrita sa mení na striktné príkazy PRIMARY KEY, FOREIGN KEY a CHECK.
- **Príprava na výkon:** Navrhujú sa indexy (INDEX) na stĺpce, cez ktoré bude aplikácia e-shopu najčastejšie vyhľadávať (napr. vyhľadávanie produktu podľa kódu alebo prihlásenie cez email).
- Všimnite si, že už nehovoríme len, že ide o 'text' alebo 'číslo'. Používame presné formáty. Napríklad pri emaille používame VARCHAR(150) s obmedzením UNIQUE, aby systém fyzicky vyhodil chybu, ak by sa dvaja ľudia pokúsili registrovať s rovnakým emailom

Architektúra fyzického dátového modelu

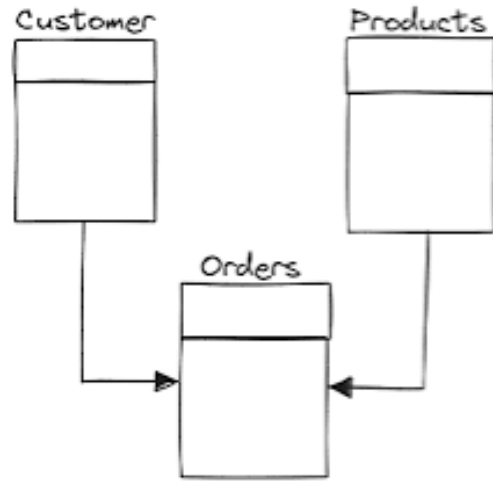
1. Reprezentácia, popis dát – štruktúra, vzťahy napr. relačný model – tabuľky, alebo pološtruktúrované modely – stromy, XML	DDL
2. Ohraničenia – typy	
3. Operácie nad dátami	DML

ERD = Entity-Relationship Diagram

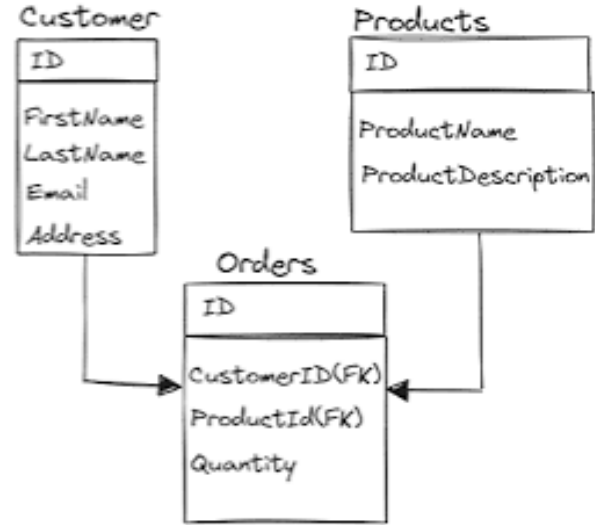


Zhrnutie

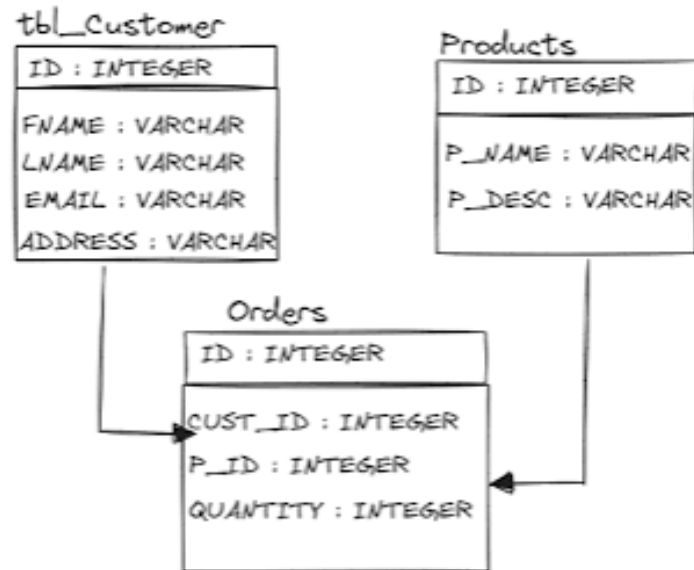
CONCEPTUAL MODEL



LOGICAL MODEL



PHYSICAL MODEL



Zhrnutie

Prvok (Element)	Konceptuálny model	Logický model	Fyzický model
Názvy entít	Áno	Áno	Nie (Menia sa na tabuľky)
Vzt'ahy medzi entitami	Áno	Áno	Nie (Menia sa na cudzie kľúče)
Atribúty	Len hlavné	Áno	Nie (Menia sa na stĺpce)
Normalizácia	Nie	Áno	Nie (Už je aplikovaná)
Primárne kľúče	Nie	Áno	Áno
Cudzie kľúče	Nie	Áno	Áno
Názvy tabuliek	Nie	Nie	Áno
Názvy stĺpcov	Nie	Nie	Áno
Dátové typy stĺpcov	Nie	Nie	Áno