1 Database design : normalization

Goals of a good data structure design

- **Data Integrity**: consistency, accuracy, avoiding anomalies
- **Query performance**: fast data retrieval
- Allow for future **expansion** of data types or relationships, evolution business requirements
- **Scalability** : growth in data volume
- **Storage** : minimize data redundancy (also important for integrity)

and : - **Simplicity**: Create an understandable structure for developers and analysts
It always comes down to balancing between read and write performance
How to design the data structure of a database ?

2 Some background and recap

quick recap - Entity-Relationship Diagrams : ERDs - 1-1, 1-many, many-many relations

and then - OLAP vs OLTP databases and design strategies - Normalization - anomalies to detect the need for normalization - normalization criteria: normal forms: 1NF, 2NF, 3NF - Denormalization when needed (OLAP) - Functional dependency

Practice:

- we'll normalize the trees v01 database

---

3 Entity Relation Diagrams

A dude called Peter Chen developed the ER diagram in 1976.

The ER model was created to visualize data structures and relationships in many situations.

- Object-Oriented Systems
- software architecture

- Business Process Modeling
- data flow in various systems
- relational databases

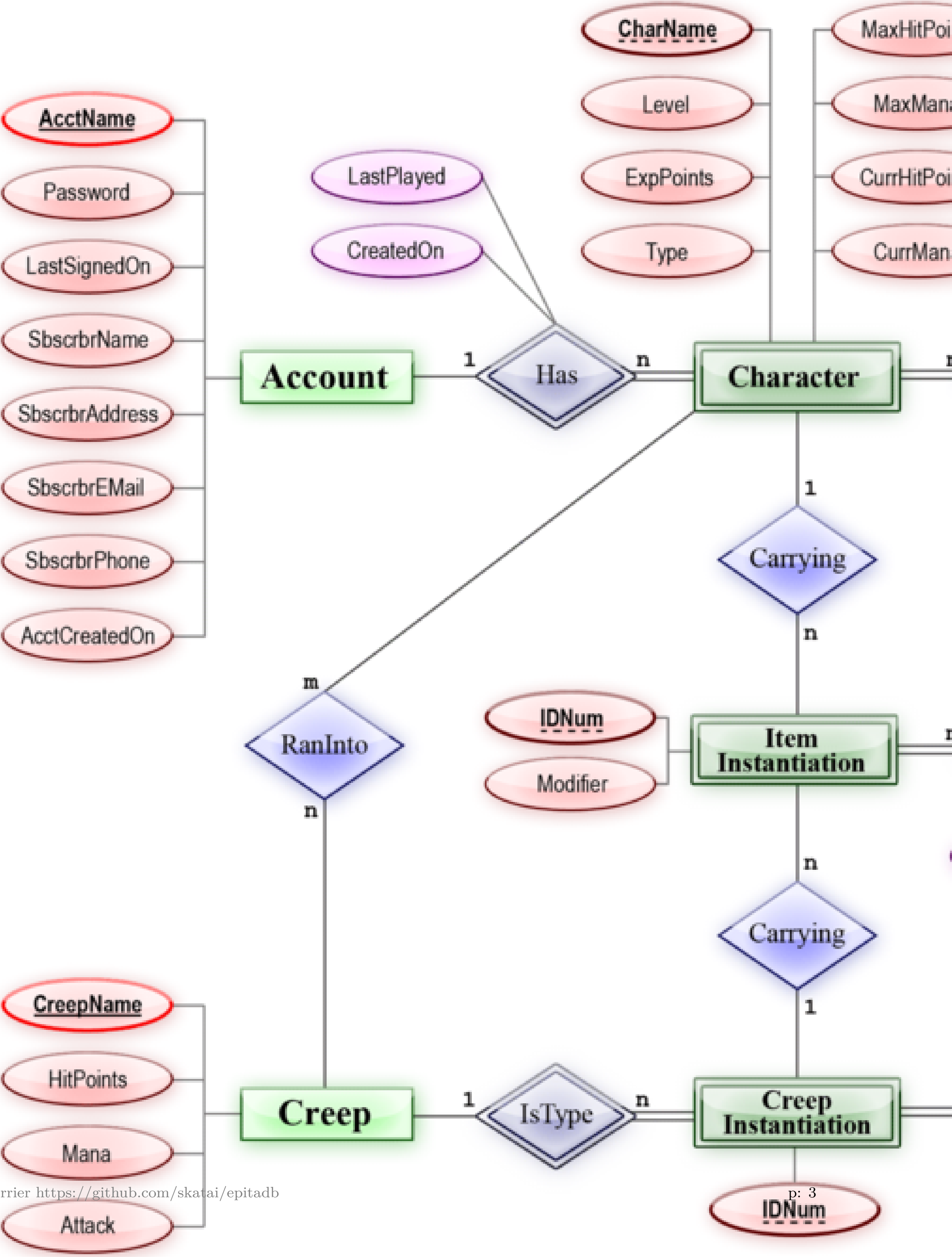
ER diagrams help in system design, information architecture, and application workflows.
The components of an ER diagram are :

- entities (tasks, real world object, ...)
- attributes
- relations between the entities

See this article for a complete explanation of ER diagrams. As you can see there are many types of entities and attributes : strong, weak, key, composite, etc ...

Introduction of ER model

see also the wikipedia page


CharName
MaxHitPoi
Level
MaxMana
ExpPoints
CurrHitPoi
Type
CurrMana
LastPlayed
CreatedOn
AcctName
Password
LastSignedOn
SbscrbrName
SbscrbrAddress
SbscrbrEMail
SbscrbrPhone
AcctCreatedOn
Account
1
Has
n
Character
n
1
Carrying
n
m
RanInto
n
IDNum
Modifier
Item Instantiation
n
n
Carrying
1
CreepName
HitPoints
Mana
Attack
Creep
1
IsType
n
Creep Instantiation
IDNum

3.1 ER diagram for relational databases

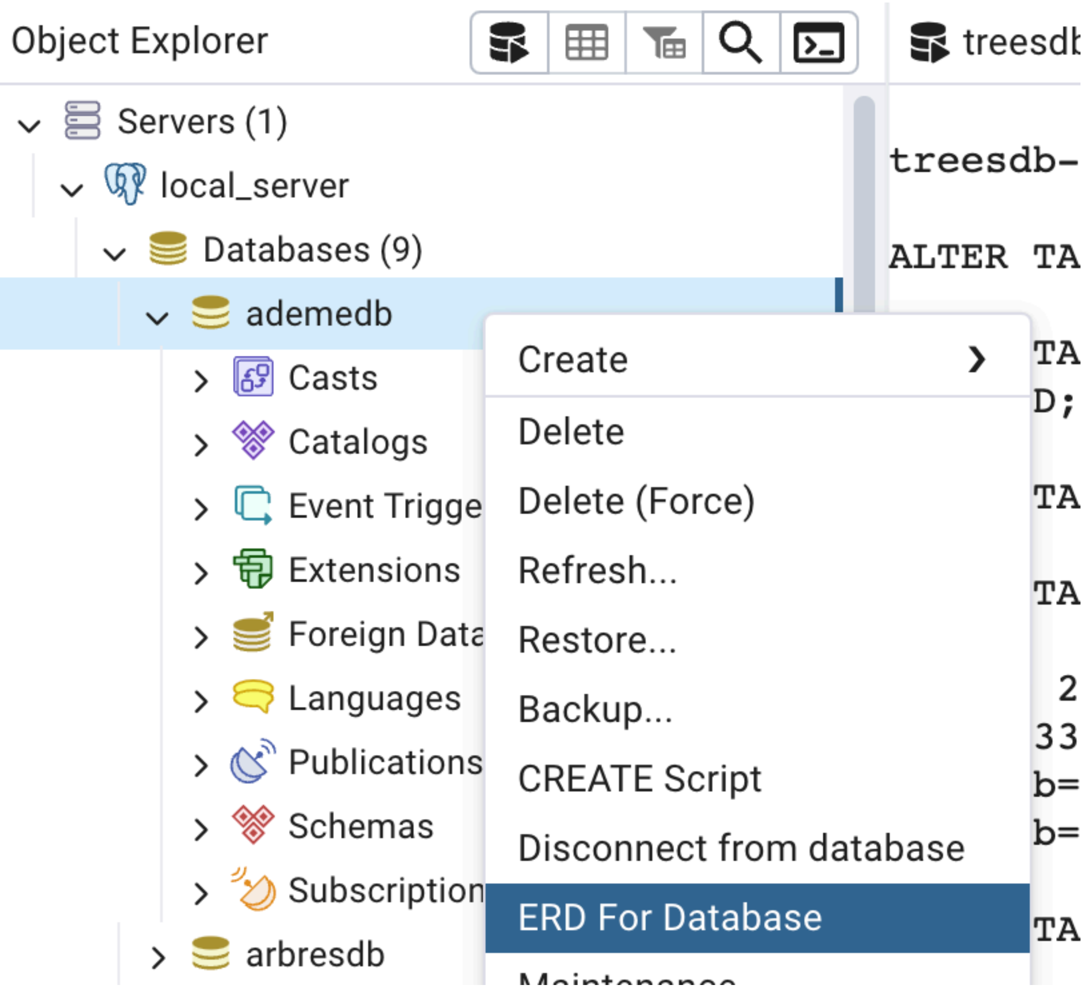
For databases, **the ER Diagram represent the structure of the database.**

- entities are **tables**
- attributes are table **columns**
- **relations** between entities can be
  - one to one
  - one to many
  - many to many

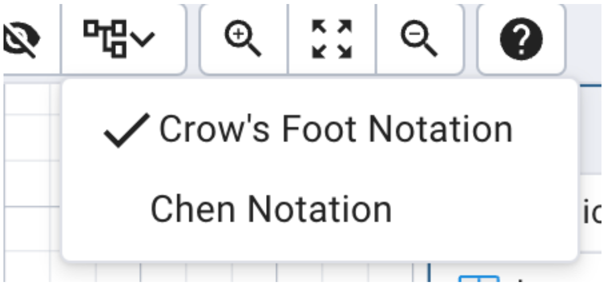
The ER diagram displays the **relations** between the **entities** (tables) present in the database and lists their **attributes** (columns)

3.2 Generate and ERD in pgAdmin

- connect to the remote server on the airdb database
- click right on the database name
- click on ERD for database



You can change notation for the relation type with



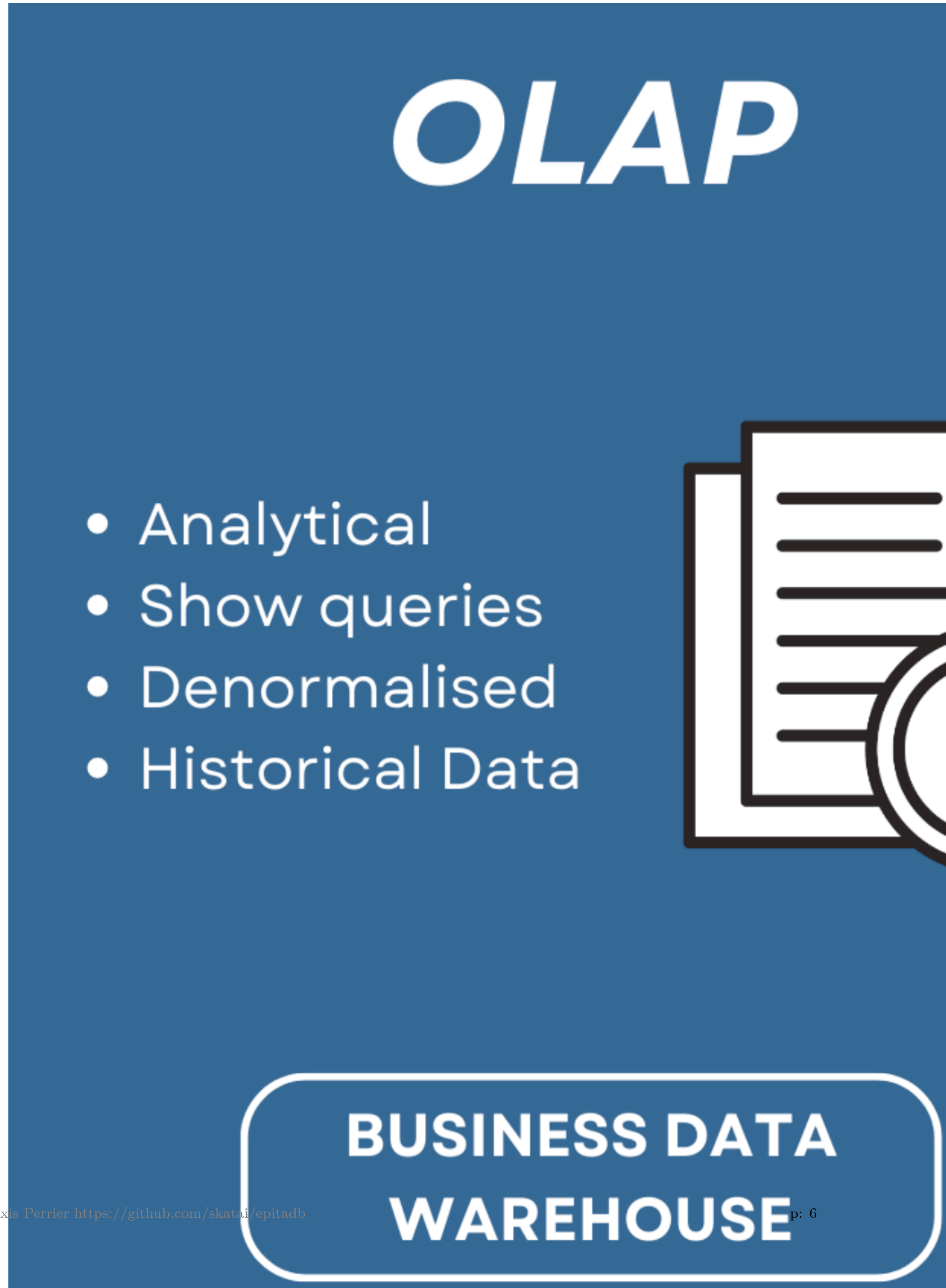
4 OLAP vs OLTP

The end usage of the database drives its data structure

analytical databases vs **transactional** databases

OLAP : Online Analytical Processing - analysis, BI, reporting, dashboards, - optimized for high read volume - complex queries (lots of joins and calculations) which have to be somewhat fast - can be asynchronous, query execution does not have to be lightning fast

OLTP: Online Transaction Processing, - applications, transactions, high write volume - optimized for high write volume: **data integrity**, fast updates and inserts - ACID properties for transactions (all or nothing) (ACID: (Atomicity, Consistency, Isolation, Durability)) - synchronous, real time



OLAP

- Analytical
- Show queries
- Denormalised
- Historical Data

BUSINESS DATA WAREHOUSE

- Further reading (look at the difference table and the Q&A at the end of the article) : difference between olap and oltp in dbms

4.1 Quiz

For each scenario, determine whether it's more suited for an OLTP (Online Transaction Processing) or OLAP (Online Analytical Processing) system.

- Liking a friend's post on Instagram.
- Analyzing trending hashtags on Twitter over the past month.
- Sending a Snapchat message to a friend.
- Netflix recommending shows based on your viewing history.
- Ordering food through a delivery app.
- Making an in-app purchase.
- TikTok or Youtube analyzing which video types keep users watching longer.
- A fitness app calculating your average daily steps for the past year.

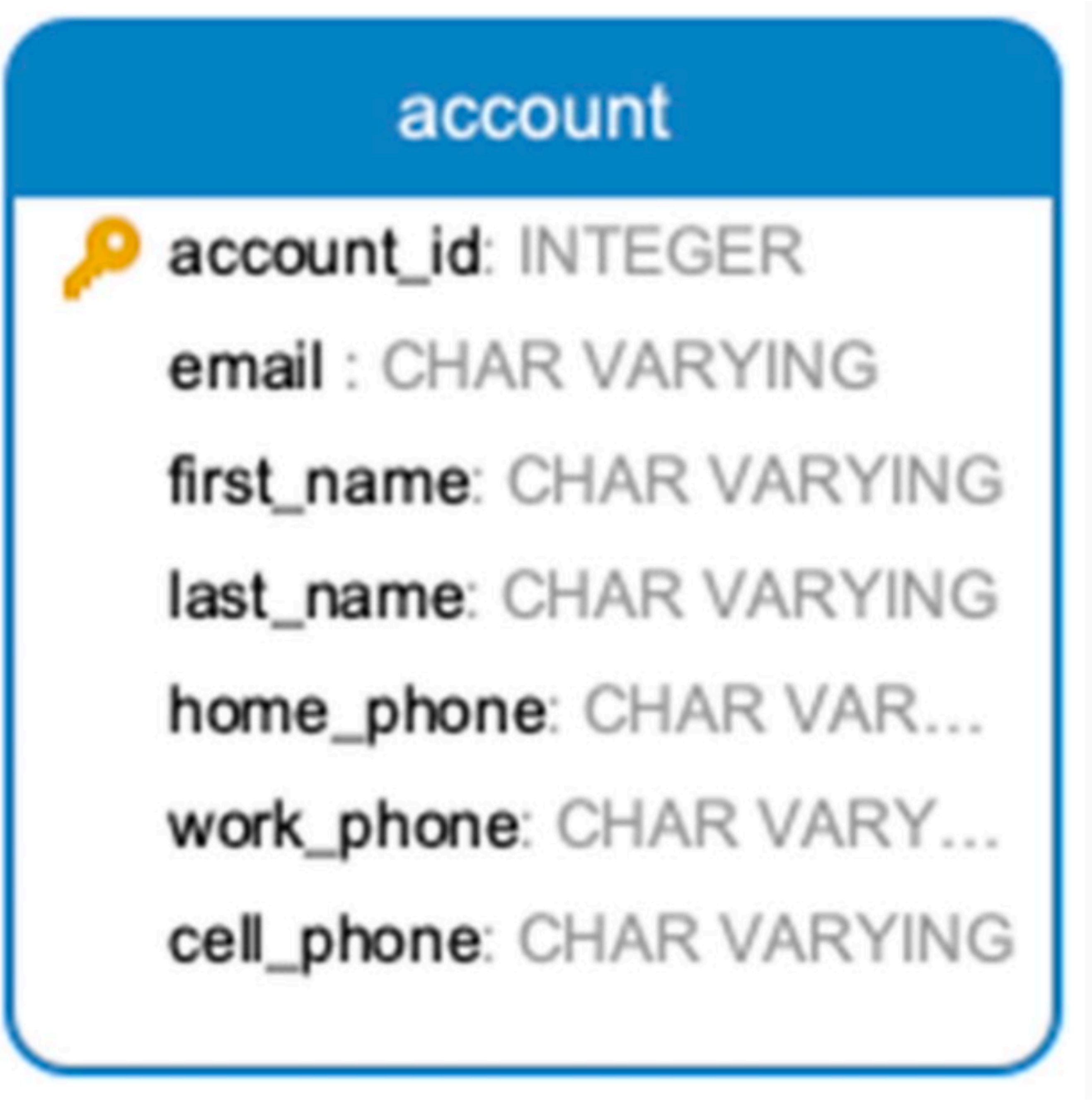
solution

your answers

---

5 Choosing between 2 designs

1 account table with multiple phones



account

account_id: INTEGER

email : CHAR VARYING

first_name: CHAR VARYING

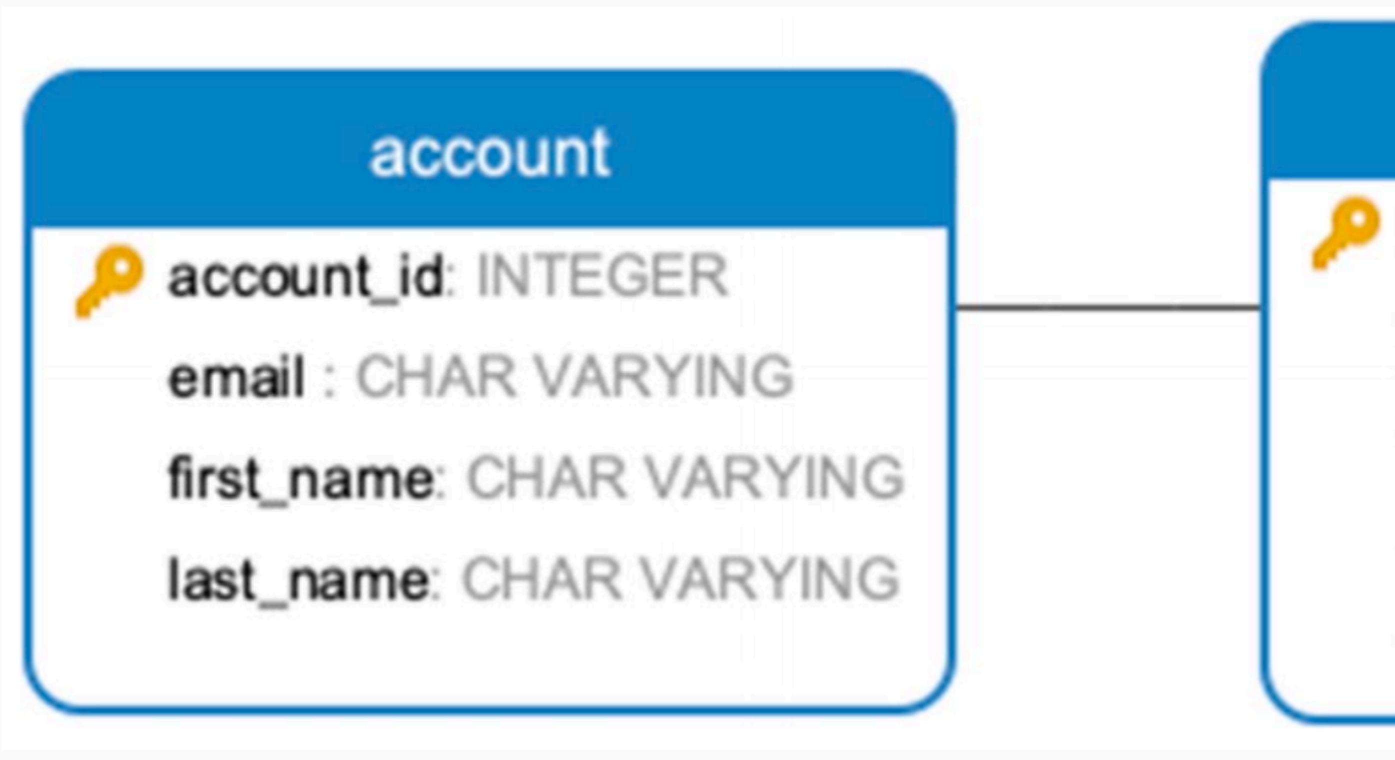
last_name: CHAR VARYING

home_phone: CHAR VAR...

work_phone: CHAR VARY...

cell_phone: CHAR VARYING

1 account table and 1 dedicated phone table



account

account_id: INTEGER

email : CHAR VARYING

first_name: CHAR VARYING

last_name: CHAR VARYING

which design (1 or 2 tables) is betterin terms of faster or simpler query for:

- fast retrieval search over phone number(s)
- dealing with missing phone type
- adding a new phone type
- flagging a phone as primary
- handling a user with no phone
- displaying all the phones of an account in a UX

---

6 Normalization

The general goal of **normalization** is to reduce data **redundancy** and **dependency** by organizing data into **separate, related tables**.

This helps maintain data integrity and flexibility:

- logical entities
- independence between tables
- uniqueness of data

Normalized databases are

- easy to update
- easy to maintain

Informally, a database is normalized if all column values depend only on the table primary key, and data is decomposed into multiple tables to avoid repetition.

In the *1 table design* for the account and its phone numbers, a phone number value depends on the name of the phone column (home_phone, work_phone, ...) not just the account_id key : it's not normalized

With a dedicated phone table, the phone value depends only on the phone_id key : normalized

7 Denormalization

The idea of denormalization is to have data redundancy to simplify queries and make OLAP queries faster

Redundant data : the same data / info exists in multiple tables

select queries may involve less joins but updates are more complex and data integrity is more complex to preserve.

7.1 Scenario:

In a social network, you have two tables:

1. **Users table**: Contains user information like `user_id`, `name`, and `email`.
2. **Posts table**: Contains posts made by users, with fields like `post_id`, `user_id`, and `content`.

In a **normalized** database: the `Posts` table only contains `user_id` as a foreign key.

If if you want to display the user's name next to their post, you need to **JOIN** `Users` and `Posts` tables.

To improve performance you can **denormalize** the Posts table by adding the `user_name` to the `Posts` table.

Denormalized `Posts` table:

	post_id	user_id	user_name	content
1	101	Ulaf	Hello world!	
2	102	Birgitte	Loving the sun	
3	103	Inge	Great day!	
4	114	Boris	When's the break?	

- Faster read performance: You can fetch the `user_name` along with the post data without needing to perform a join between the `Users` and `Posts` tables.

But

- **Data redundancy**: If Ulaf changes his name, you will need to update it in both the `Users` table and every row in the `Posts` table that references him. This increases the complexity of updates.