**Sr. Backend Developer Questions**

Q1. We have an online store with items organized in categories, sub-categories, and

sub-sub-categories up to N levels. Draw the database structure for the online store

with the category tree to store things. Each category has a name. The number and

level of categories can go up to infinite.

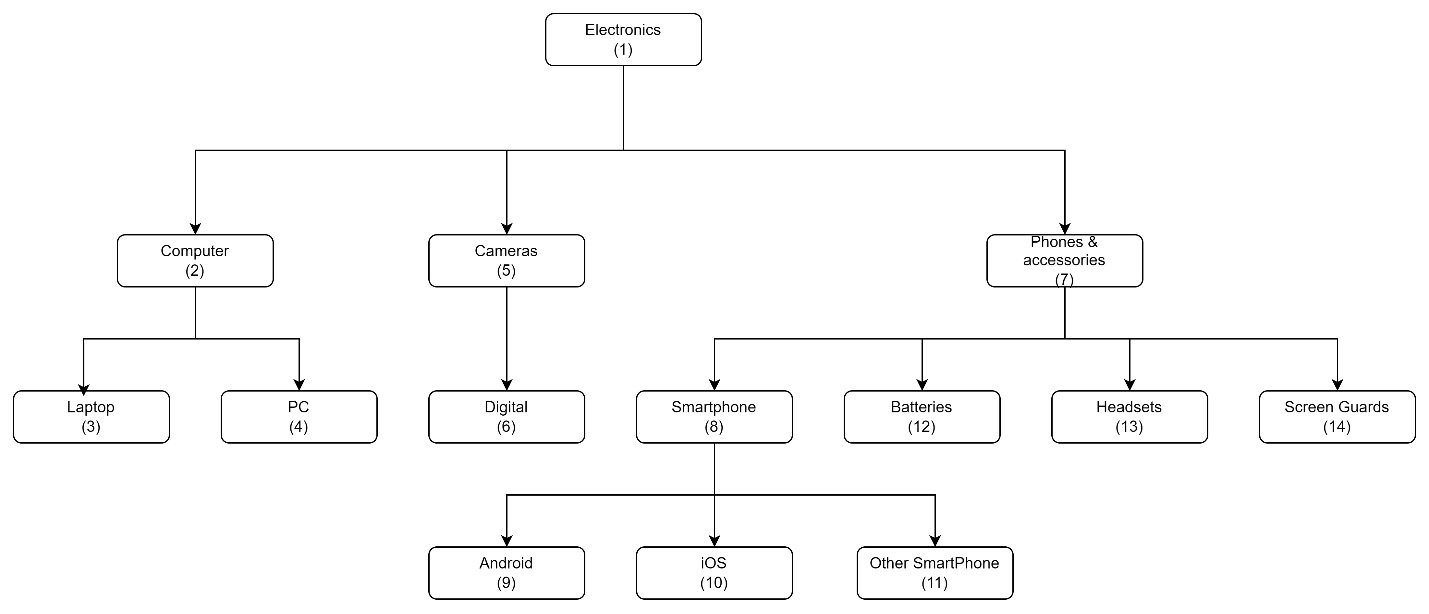
Write a query to fetch all the first-level children (categories+items) for a category.

Write a query to fetch all the children (categories+items) at all category levels for a

category.

**Q1 Answer:** Online store hierarchical data can be managed using many ways but MySQL Adjacency List Model is the simplest solution because its very simple and very popular in managing hierarchical data.

In the adjacency list model, each node has a pointer that points to its parent. The top node has no parent. See the following diagram depicts categories of electronics products:



Write a query to fetch all the first-level children (categories+items) for a category.

WITH RECURSIVE category\_path (id, title, path) AS

(

SELECT id, title, title as path FROM category

WHERE parent\_id IS NULL

UNION ALL

SELECT c.id, c.title, CONCAT(cp.path, ' > ', c.title)

FROM category\_path as cp JOIN category AS c

ON cp.id = c.parent\_id and c.parent\_id = 1

)

SELECT \* FROM category\_path ORDER BY path;



Write a query to fetch all the children (categories+items) at all category levels for a

category.

WITH RECURSIVE category\_path (id, title, path) AS

(

SELECT id, title, title as path FROM category

WHERE parent\_id IS NULL

UNION ALL

SELECT c.id, c.title, CONCAT(cp.path, ' > ', c.title)

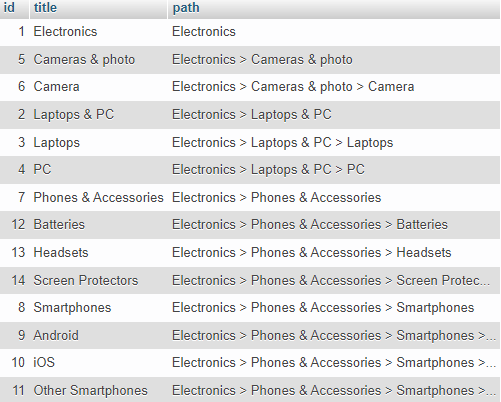
FROM category\_path as cp JOIN category AS c

ON cp.id = c.parent\_id

)

SELECT \* FROM category\_path

ORDER BY path;



Q2. Write the REST API endpoints for products consisting of (id, name, price) where

the price is in dollars.

The products now consist of (id, name, price, currency).

Define how will you apply this change, considering that

-API is in production.

-Other clients are using the existing API.

If you choose to version the API, write at a high level how the change will affect

controllers, services, entities, etc.

**Q2 Answer:** Yes, I prefer choosing versioning API since the API is in production and some clients are already using it. Assuming initial version is v1 and new version will be v1.1 since its minor change

There are multiple ways to do versioning API like below

* Versioning through URI Path. 🡺 http://www.example.com/api/v**1**/products
* Versioning through query parameters. 🡺 http://www.example.com/api/products?version=1
* Versioning through custom headers. 🡺 curl -H “Accepts-version: 1.0”  
  http://www.example.com/api/products
* Versioning through content negotiation. 🡺 curl -H “Accept: application/vnd.xm.device+json; **version=1**” <http://www.example.com/api/products>

This change will affect controller because it contains the flow control logic for an MVC application. A controller determines what response to send back to a user when a user makes a browser request based on API version user sent with request else it has to fall in default version that’s v1 to get the appropriate API data

API services are **a means for an application to interact with a server-side system to retrieve and/or update data**.

This change will affect Entities **because these** are data structures that can be associated as an instance of a transaction.