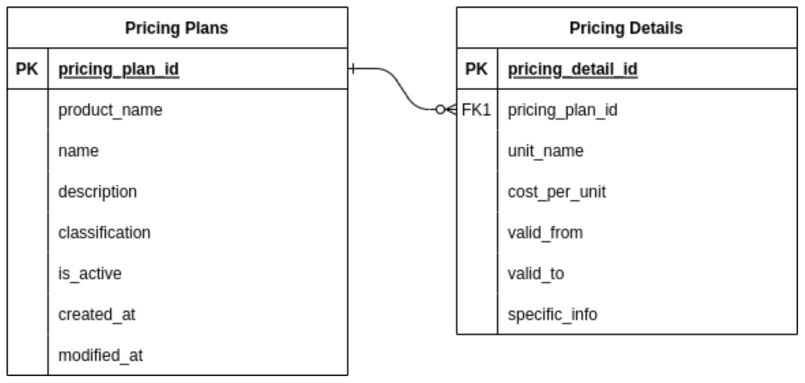
2023-08-03 Integration

DB model



Example

Pricing Plans Table

pricing_plan_id	product_name	name	description	classification	is_active	created_at	modified_at
1	sim4life	Dynamic	…	TIER	true	…	…
2	sim4life	Computational Type A	…	TIER	true	…	…
3	sim4life	Computational Type B	…	TIER	true	…	…
4	tip	CPU hours	…	CPU_HOUR	true	…	…
5	tip	Storage	…	STORAGE	true	…	…

Pricing Details Table

pricing_detail_id	pricing_paln_id	unit_name	cost_per_unit	valid_from	valid_to	specific_info
1	1	S	5	…	NULL	{“aws_instance_type“: “EC2-small“}
2	1	M	10	…	NULL	{“aws_instance_type“: “EC2-medium“}
3	1	L	15	…	NULL	{“aws_instance_type“: “EC2-large“}
4	2	S	10	…	NULL	{…}

5	2	M	20	…	NULL	{…}
6	2	L	30	…	NULL	{…}
7	3	XXL	50	…	NULL	{…}
8	4	CPU Hours	3	…	NULL	NULL
9	5	Storage	10	…	NULL	NULL

Integration

GET /pricing-plans --> Retrieves a list of all active pricing plans for the product

GET /pricing-plans/{id}/pricing-details → Retrieves all pricing details associated with a specific pricing plan. This includes mainly unit_name, cost_per_unit.

When the client is starting a job with something like **POST /run** he should provide us with:

{ "pricing_plan_id": 1, "pricing_detail_id": 3 }

By using IDs instead of names we make it more rigid. For example, these ids, similarily to wallet_id can be passed to the resource usage tracker. What is good with this approach is that this id uniquely defines the cost_per_unit at that moment, so we can always easily reconstruct the cost.

Also if we would rename the “M” to “Medium” we would not break the API.