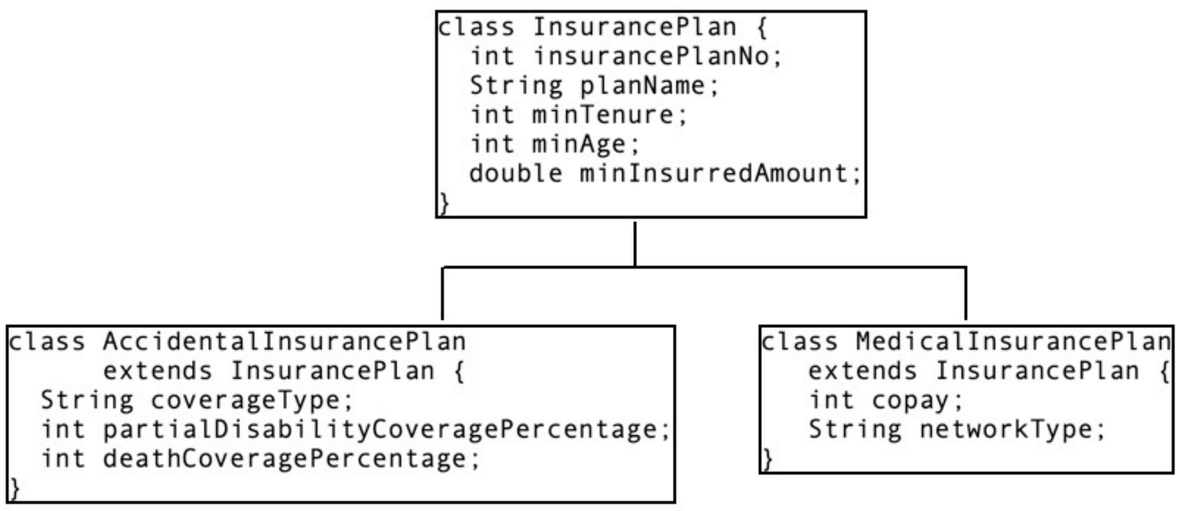
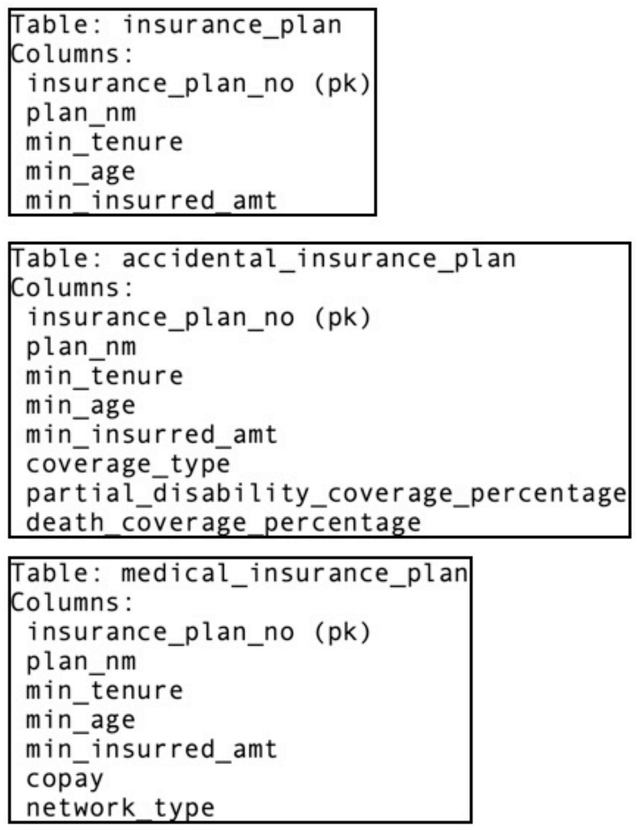
Inheritance Mapping Model: Table per concrete class
Jpa: TABLE_PER_CLASS





How to perform persistence operations for these hierarchy of classes into the underlying tables based on the inheritance strategy: Table per concrete class, let us explore:

1. persist
1.1 How to persist superclass object into the underlying table model (InsurancePlan):
directly store the data into insurance_plan table

1.2 How to persist the subclass object into the underlying table model (AccidentalInsurancePlan/MedicalInsurancePlan):
since each class in the hierarchy has separate tables independently, we can store each of the subclass objects into their independent tables directly along with inherited attributes also

AccidentalInsurancePlan (object) = accidental_insurance_plan (table)
MedicalInsurancePlan (object) = medical_insurance_plan (table)

Important:
Even though for all the hierarchy of entity classes, we have separate (independent) tables with individual primary keys, to support polymorphic retrieval of the data we should ensure the pk column value across the entities/tables of the hierarchy is unique (not individual), so that we can easily identify the record we fetched is of which classType object of the hierarchy. Because with the given id, the record can exist only in one of the tables of the hierarchy, so based on which table the record exists we can identify the classType. Now we can support polymorphic access.

Note: when working with TABLE_PER_CLASS / Table per concrete class: we should not use IDENTITY as an id generator.

---

2. Fetch / Get
2.1 How to fetch the Subclass object from the underlying table (AccidentalInsurancePlan/MedicalInsurancePlan):

session.get(AccidentalInsurancePlan.class, 2); (or) session.get(MedicalInsurancePlan.class, 3);

directly we can goto the respective tables of the corresponding subclasses and can query the data based on the primary key column. if the record exists within that table, return the object otherwise return null

2.2 How to fetch the superclass object from the underlying table (InsurancePlan.class):
session.get(InsurancePlan.class, 1);

To support polymorphic retrieval (any of the subclass objects can be queries based on superclass type) we ensured the id of the entity classes across the tables is unique.
So when we are querying the data for a superclass type. we need to fetch the data from all the #3 tables of the hierarchy and see the record was found in which table

1. if it exists only in insurance_plan table = InsurancePlan (object)
2. if it exists only in accidental_insurance_plan table = AccidentalInsurancePlan (object)
3. if it exists only in medical_insurance_plan table = MedicalInsurancePlan (object)

note: no #2 tables has a record with same primary key value.

How to query the data from #3 tables?
we need to use union query. union means all the query outputs are unioned and returned as one output.

```
select * from
(select ip.insurance_plan_no, ip.plan_nm, ip.min_tenure, ip.min_age, ip.min_insurred_amt, null as coverage_type, null as dcp, null as dcp, null as copay,
null as network_type, '0' as clazz from insurance_plan ip
union
select aip.insurance_plan_no, aip.plan_nm, aip.min_tenure, aip.min_age, aip.min_insurred_amt, aip.coverage_type, aip.pdcp, aip.dcp, null as copay, null as
network_type, '1' as clazz from accidental_insurance_plan aip
union
select mip.insurance_plan_no, mip.plan_nm, mip.min_tenure, mip.min_age, mip.min_insurred_amt, null as coverage_type, null as dcp, null as dcp, mip.copay,
mip.network_type, '2' as clazz from medical_insurance_plan mip) where insurance_plan_no = 2;
```