```python
from trytond.model import ModelSQL, ModelView, fields
from trytond.pool import PoolMeta

__all__ = ['Unit', 'Measurement', 'ItemBreakDown', 'Item', 'Chapter', 'Budget', 'Project']
__metaclass__ = PoolMeta

# ------------------
# UNIT
# ------------------
class Unit(ModelSQL, ModelView):
    'Unit'
    __name__ = 'proyects.unit'
    __bc3_record__ = '~U'

    ID = fields.Char('ID', required=True)
    Code = fields.Char('Code', required=True)
    Description = fields.Text('Description')


# ------------------
# MEASUREMENT
# ------------------
class Measurement(ModelSQL, ModelView):
    'Measurement'
    __name__ = 'proyects.measurement'
    __bc3_record__ = '~L'

    ID = fields.Char('ID', required=True)
    Expression = fields.Text('Expression', required=True)
    Quantity = fields.Float('Quantity', required=True)
    Item = fields.Many2One('proyects.item', 'Item', required=True)


# ------------------
# ITEM BREAKDOWN
# ------------------
class ItemBreakDown(ModelSQL, ModelView):
    'Item Breakdown'
    __name__ = 'proyects.item_breakdown'
    __bc3_record__ = '~D'

    ID = fields.Char('ID', required=True)
    Description = fields.Text('Description')
    Item = fields.Many2One('proyects.item', 'Parent Item', required=True)
    Component = fields.Many2One('proyects.item', 'Component Item', required=True)
    Quantity = fields.Float('Quantity', required=True)
    Price = fields.Numeric('Price', digits=(16, 2))
    Performance = fields.Numeric('Performance', digits=(16, 2))
    Family = fields.Selection([
        ('material', 'Material'),
        ('labor', 'Labor'),
        ('equipment', 'Equipment'),
        ('service', 'Service'),
        ('other', 'Other'),
    ], 'Family')


# ------------------
# ITEM
# ------------------
class Item(ModelSQL, ModelView):
    'Item'
    __name__ = 'proyects.item'
    __bc3_record__ = '~P'

    ID = fields.Char('ID', required=True)
```

```
    Code = fields.Char('Code', required=True)
    Name = fields.Char('Name', required=True)
    Description = fields.Text('Description')

    Quantity = fields.Float('Quantity', required=True)
    Unit = fields.Many2One('proyects.unit', 'Unit', required=True)
    Chapter = fields.Many2One('proyects.chapter', 'Chapter', required=True)

    VAT = fields.Numeric(
        'VAT',
        digits=(5, 2),
        help='Specific VAT for this item. Overrides the budget VAT if set.'
    )

    Breakdown = fields.One2Many('proyects.item_breakdown', 'Item', 'Breakdown')
    Measurements = fields.One2Many('proyects.measurement', 'Item', 'Measurements')
    Price = fields.Numeric('Price', digits=(16, 2))

    # Computed fields
    CompoundPrice = fields.Function(fields.Numeric('Compound Price', digits=(16,2)),
'get_compound_price')
    TotalCost = fields.Function(fields.Numeric('Total Cost', digits=(16,2)),
'get_total_cost')
    CostWithVAT = fields.Function(fields.Numeric('Cost with VAT', digits=(16,2)),
'get_cost_with_vat')

    @classmethod
    def get_compound_price(cls, items, names):
        result = {}
        for item in items:
            subtotal = 0
            for bd in item.Breakdown:
                qty = bd.Quantity or 0
                price = bd.Price or 0
                subtotal += qty * price
            result[item.id] = subtotal
        return result

    @classmethod
    def get_total_cost(cls, items, names):
        result = {}
        for item in items:
            price = item.CompoundPrice or item.Price or 0
            result[item.id] = (item.Quantity or 0) * price
        return result

    @classmethod
    def get_cost_with_vat(cls, items, names):
        result = {}
        for item in items:
            base_cost = item.TotalCost or 0
            vat = item.VAT if item.VAT is not None else (item.Chapter.Budget.VAT if
item.Chapter and item.Chapter.Budget else 0)
            result[item.id] = base_cost * (1 + (vat or 0)/100)
        return result


# ------------------
# CHAPTER
# ------------------
class Chapter(ModelSQL, ModelView):
    'Chapter'
    __name__ = 'proyects.chapter'
    __bc3_record__ = '~C'

    ID = fields.Char('ID', required=True)
```

```
    Code = fields.Char('Code', required=True)
    Name = fields.Char('Name', required=True)
    Description = fields.Text('Description')

    Budget = fields.Many2One('proyects.budget', 'Budget', required=True)
    Parent = fields.Many2One('proyects.chapter', 'Parent Chapter')
    Children = fields.One2Many('proyects.chapter', 'Parent', 'Children')
    Items = fields.One2Many('proyects.item', 'Chapter', 'Items')


# ------------------
# BUDGET
# ------------------
class Budget(ModelSQL, ModelView):
    'Budget'
    __name__ = 'proyects.budget'
    __bc3_record__ = '~K'

    ID = fields.Char('ID', required=True)
    Name = fields.Char('Name', required=True)
    Project = fields.Many2One('proyects.project', 'Project', required=True)
    Date = fields.Date('Date', required=True)
    Status = fields.Selection([
        ('draft', 'Draft'),
        ('confirmed', 'Confirmed'),
        ('approved', 'Approved'),
        ('cancelled', 'Cancelled')
    ], 'Status', required=True)

    Chapters = fields.One2Many('proyects.chapter', 'Budget', 'Chapters')
    IndirectCostsRate = fields.Numeric('Indirect Costs Rate', digits=(5,2))
    Discount = fields.Numeric('Discount', digits=(16,2))
    VAT = fields.Numeric('Default VAT', digits=(5,2),
                         help='Default VAT for items. Items can override this value.')

    TotalCost = fields.Function(fields.Numeric('Total Cost', digits=(16,2)),
'get_total_cost')
    TotalCostWithVAT = fields.Function(fields.Numeric('Total Cost with VAT',
digits=(16,2)), 'get_total_cost_with_vat')

    @classmethod
    def get_total_cost(cls, budgets, names):
        result = {}
        for budget in budgets:
            total = 0
            for chapter in budget.Chapters:
                for item in chapter.Items:
                    total += item.TotalCost or 0
            result[budget.id] = total
        return result

    @classmethod
    def get_total_cost_with_vat(cls, budgets, names):
        result = {}
        for budget in budgets:
            total = 0
            for chapter in budget.Chapters:
                for item in chapter.Items:
                    vat = item.VAT if item.VAT is not None else (budget.VAT or 0)
                    total += (item.TotalCost or 0) * (1 + vat/100)
            result[budget.id] = total
        return result


# ------------------
# PROJECT
```

```python
# ------------------
class Project(ModelSQL, ModelView):
    'Project'
    __name__ = 'proyects.project'
    __bc3_record__ = '~K'

    ID = fields.Char('ID', required=True)
    Name = fields.Char('Name', required=True)
    Description = fields.Text('Description')

    Customer = fields.Many2One('res.party', 'Customer', required=True)
    Author = fields.Char('Author')
    Company = fields.Char('Company')
    VersionBC3 = fields.Char('BC3 Version')

    Budgets = fields.One2Many('proyects.budget', 'Project', 'Budgets')
```