Tentu! Untuk membangun sistem database komunitas marga dengan fokus pada manajemen anggota dan visualisasi tarombo, berikut adalah **daftar modul Odoo yang perlu diinstall** beserta penjelasan mengapa modul tersebut diperlukan.

Modul-modul ini akan menjadi fondasi yang kuat sebelum Anda mengembangkan modul kustom (batak\_community).

Daftar Modul yang Direkomendasikan

| **No.** | **Nama Modul (ID Teknis)** | **Kategori** | **Pentingnya** |
| --- | --- | --- | --- |
| 1 | **Contacts** (contacts) | Sales | **Wajib**. Modul inti untuk mengelola model res.partner (anggota komunitas). |
| 2 | **Website** (website) | Website | **Sangat Disarankan**. Untuk membangun portal anggota tempat mereka bisa login dan melihat data tarombo mereka sendiri. |
| 3 | **Accounting** (account) | Accounting | **Sangat Disarankan**. Untuk mengelola keuangan komunitas, iuran, dan donasi dengan transparan. |
| 4 | **Events** (event) | Marketing | **Disarankan**. Untuk mengelola acara-acara adat (Pesta, Mangongkal Holi, dll). |
| 5 | **Discuss** (mail) | Discuss | **Otomatis Terinstall**. Inti untuk sistem notifikasi, chatter, dan komunikasi internal. |
| 6 | **SMS** (sms) | Discuss | **Opsional**. Jika ingin mengirimkan notifikasi atau pengingat iuran via SMS. |

Penjelasan Detail dan Alasan Install

1. Contacts (contacts)

* **Fungsi:** Modul ini adalah tulang punggung dari sistem Anda. Ini memperluas modul dasar res.partner untuk memberikan antarmuka yang user-friendly dalam mengelola kontak (anggota komunitas).
* **Mengapa Diperlukan:** Semua field kustom yang Anda buat (seperti marga, father\_id) akan ditambahkan ke model res.partner dan akan muncul secara native di dalam aplikasi **Contacts**. Tanpa modul ini, Anda harus mengelola kontak dari menu teknis yang kurang user-friendly.

2. Website (website)

* **Fungsi:** Memungkinkan Anda membuat dan mengelola website secara visual. Fitur terpentingnya adalah **Portal**.
* **Mengapa Diperlukan:** Anda akan membuat halaman portal khusus anggota yang aman. Di sini, anggota dapat:
  + Login dengan akun mereka.
  + Melihat dan memperbarui profil mereka sendiri.
  + **Melihat visualisasi tarombo dari keluarganya** (ini adalah tujuan utama Anda).
  + Melihat status keuangan/iuran mereka.
  + Mendaftar untuk acara.
* **Cara Kerja:** Modul kustom Anda (batak\_community) akan membuat controller dan template halaman web khusus yang hanya bisa diakses oleh anggota yang login.

3. Accounting (account)

* **Fungsi:** Modul akuntansi lengkap untuk mengelola pembukuan.
* **Mengapa Diperlukan:** Meskipun fokus Anda adalah database, komunitas hampir pasti punya aspek keuangan (iuran bulanan/tahunan, sumbangan acara, dll.). Modul ini memungkinkan Anda:
  + Membuat **Product** bernama "Iuran Bulanan Jan 2024", "Sumbangan Pesta", dll.
  + Membuat **Invoice** (tagihan) otomatis untuk anggota.
  + Mencatat pembayaran yang masuk.
  + Menghasilkan laporan keuangan yang jelas dan profesional untuk bendahara.
* **Integrasi:** Data anggota dari **Contacts** akan otomatis menjadi pelanggan (customer) yang bisa ditagih.

4. Events (event)

* **Fungsi:** Untuk mengelola acara dari perencanaan hingga eksekusi.
* **Mengapa Disarankan:** Sangat relevan dengan budaya Batak yang kaya akan acara adat. Anda dapat:
  + Membuat acara "Pesta Pernikahan Arian Sitompul".
  + Mengelola daftar undangan (yang berasal dari database anggota).
  + Mengelola daftar hadir dan konfirmasi kehadiran.
  + Menggunakan **Website** module untuk membuat halaman pendaftaran acara.

5. Discuss (mail)

* **Fungsi:** Inti untuk semua jenis komunikasi dalam Odoo.
* **Mengapa Diperlukan:** Modul ini akan terinstall secara otomatis. Ia menambahkan "chatter" (seperti feed diskusi) di setiap record (anggota, acara, invoice). Ini berguna untuk mencatat histori komunikasi, misalnya: "Sudah dihubungi via telepon pada tanggal X untuk konfirmasi data ayah."

6. SMS (sms)

* **Fungsi:** Mengirim dan menerima SMS langsung dari Odoo.
* **Mengapa Opsional:** Jika anggota komunitas lebih responsif via SMS atau WhatsApp (integrasi WhatsApp membutuhkan usaha lebih), modul SMS bisa digunakan untuk mengirim pengingat iuran atau undangan acara. Ini membutuhkan kredit SMS dari penyedia layanan.

Ringkasan Langkah Install

1. **Install Odoo** (Community Edition sudah cukup).
2. Masuk sebagai **Administrator**.
3. Masuk ke Apps menu, **cari dan install modul-modul berikut**:
   * Contacts
   * Website  
     \*Accounting (biasanya sudah terinstall sebagian, pastikan lengkap)
   * Events (jika diperlukan)
   * SMS (jika diperlukan)
4. **Develop & Install Modul Kustom** Anda (batak\_community) yang berisi:
   * Field-field kustom pada res.partner (marga, father\_id, dll.)
   * View kustom untuk MacMillan field-field tersebut.
   * Controller dan template untuk halaman **Tarombo Viewer**.
   * Logic untuk menghitung generasi (generation).

Dengan modul-modul inti ini terinstall, Anda memiliki platform yang sangat kuat untuk mulai mengembangkan fitur-fitur spesifik budaya Batak di atasnya. **Horas!**

Excellent! Fokus pada database komunitas dan visualisasi Tarombo adalah fondasi yang sangat tepat. Berikut adalah panduan mendetail untuk mendesain sistem database dan tampilan tree Tarombo dalam Odoo.

**Analisis Khusus: Modul Database & Tarombo untuk Komunitas Marga**

Konsep kunci yang akan kita gunakan adalah **rekursi**—setiap anggota (res.partner) memiliki referensi ke ayahnya (father\_id), yang juga merupakan seorang res.partner. Dari satu record "leluhur tertinggi", kita dapat menelusuri semua keturunannya.

**1. Desain Model Data (Custom Module:**batak\_community**)**

Langkah pertama adalah membuat modul Odoo kustom, misalnya bernama batak\_community. Modul ini akan meng-*extend* model res.partner.

**Field Kustom yang akan Ditambahkan ke**res.partner**:**

| **Field Name (Technical)** | **Tipe Field** | **Deskripsi** |
| --- | --- | --- |
| marga | Char | Nama marga (Contoh: Simatupang, Sitompul). Bisa juga dibuat Selection jika daftar marga sudah pasti. |
| sub\_marga | Char | (Opsional) Sub-marga atau marga cabang. |
| father\_id | Many2one('res.partner') | **Field paling penting.** Menunjuk ke record ayah dari anggota ini. Ini yang membentuk hierarki. |
| mother\_id | Many2one('res.partner') | (Opsional) Menunjuk ke record ibu. |
| generation | Integer | Generasi ke- dari leluhur paling tua. Leluhur = 0, anaknya = 1, cucunya = 2, dst. |
| is\_anggota\_aktif | Boolean | Status keanggotaan. |
| tanggal\_lahir | Date | Untuk menghitung umur dan urutan anak. |
| tempat\_lahir | Char | Daerah kelahiran. |
| jenis\_kelamin | Selection | ([('male', 'Laki-laki'), ('female', 'Perempuan')]) |

**File:**models/models.py

python

from odoo import models, fields, api

class ResPartner(models.Model):

\_inherit = 'res.partner'

marga = fields.Char(string='Marga')

sub\_marga = fields.Char(string='Sub-Marga')

father\_id = fields.Many2one('res.partner', string='Ayah')

mother\_id = fields.Many2one('res.partner', string='Ibu')

generation = fields.Integer(string='Generasi', compute='\_compute\_generation', store=True)

is\_anggota\_aktif = fields.Boolean(string='Anggota Aktif', default=True)

tanggal\_lahir = fields.Date(string='Tanggal Lahir')

tempat\_lahir = fields.Char(string='Tempat Lahir')

jenis\_kelamin = fields.Selection([('male', 'Laki-laki'), ('female', 'Perempuan')], string='Jenis Kelamin')

*# Fungsi untuk otomatis menghitung generasi*

@api.depends('father\_id.generation')

def \_compute\_generation(self):

for record in self:

if record.father\_id:

record.generation = record.father\_id.generation + 1

else:

*# Jika tidak ada ayah yang ditentukan, asumsikan dia adalah leluhur (generasi 0)*

record.generation = 0

**2. Membangun Tampilan Tree (Tarombo Viewer)**

Ini adalah bagian yang paling menantang. Odoo's standard tree view tidak dirancang untuk menangani hierarki yang dalam dan kompleks seperti tarombo. Kita memerlukan solusi front-end yang powerful.

**Opsi 1: Menggunakan Widget Hierarchy Odoo (Terbatas)**  
Odoo memiliki widget hierarchy yang bisa digunakan di views. Ini sederhana tetapi mungkin tidak cukup untuk menampilkan seluruh marga.

**XML View (**views/partner\_views.xml**)**

xml

<record id="view\_partner\_tree\_inherited" model="ir.ui.view">

<field name="name">res.partner.tree.inherited</field>

<field name="model">res.partner</field>

<field name="inherit\_id" ref="base.view\_partner\_tree"/>

<field name="arch" type="xml">

<xpath expr="//field[@name='name']" position="after">

<field name="marga"/>

<field name="father\_id" widget="hierarchy"/>

<field name="generation"/>

</xpath>

</field>

</record>

**Opsi 2: Menggunakan Library JavaScript External (Rekomendasi)**  
Ini adalah solusi terbaik. Kita akan membuat view kustom yang mengintegrasikan library graphing seperti **D3.js** atau **vis.js**.

**Langkah-langkah:**

1. **Buat Controller HTTP:** Buat sebuah route di Odoo yang mengembalikan data JSON berisi struktur nodes (anggota) dan edges (hubungan ayah-anak) untuk visualisasi.
2. **Buat Template QWeb:** Buat halaman web baru yang menyertakan library JS dan <div> container untuk graph.
3. **Tulis JavaScript:** Tulis kode JS yang mengambil data JSON dari controller dan merender graph yang interaktif.

**Contoh Sederhana Controller (**controllers/controllers.py**):**

python

from odoo import http

from odoo.http import request

import json

class TaromboController(http.Controller):

@http.route('/batak\_community/tarombo/json/<int:ancestor\_id>', auth='user')

def get\_tarombo\_json(self, ancestor\_id, \*\*kwargs):

*# Dapatkan anggota starting dari leluhur tertentu*

ancestor = request.env['res.partner'].browse(ancestor\_id)

nodes = []

edges = []

def get\_descendants(partner, level=0):

nodes.append({

'id': partner.id,

'label': partner.name + " " + (partner.marga or ''),

'generation': partner.generation

})

children = request.env['res.partner'].search([('father\_id', '=', partner.id)])

for child in children:

edges.append({

'from': partner.id,

'to': child.id

})

get\_descendants(child, level + 1) *# Rekursi*

*# Jalankan fungsi, dimulai dari leluhur*

get\_descendants(ancestor)

data = {'nodes': nodes, 'edges': edges}

return request.make\_response(json.dumps(data), headers=[('Content-Type', 'application/json')])

**Kode JS (dalam template) akan menggunakan D3.js untuk membuat SVG yang menampilkan nodes dan links.** (Kode D3.js cukup panjang dan kompleks, tetapi banyak contoh online untuk "collapsible tree").

**3. UI untuk Memilih Leluhur & Melihat Tree**

1. **Buat Action dan Menu Baru:**
   * Buat sebuah menu item "Tarombo Viewer" yang akan membuka halaman kustom.
2. **Buat Form untuk Memilih Starting Point:**
   * Sebelum menampilkan tree, buat form sederhana dimana user bisa memilih dari res.partner mana (leluhur mana) mereka ingin mulai menelusuri tarombo.
3. **Integrasikan dengan Template:**
   * Form tersebut akan mengarahkan ke URL template graph dengan parameter ID leluhur yang dipilih. Template graph akan memanggil controller /batak\_community/tarombo/json/<ID> untuk mendapatkan data dan merendernya.

**4. Langkah Implementasi Praktis (Step-by-Step)**

1. **Buat Modul Baru:** ./odoo-bin scaffold batak\_community .
2. **Define Model:** Edit models/models.py untuk meng-*inherit* res.partner dan tambahkan field seperti kode di atas.
3. **Define Views:** Edit views/partner\_views.xml untuk menambahkan field-field kustom ke form view dan tree view standar partner.
4. **Buat Controller:** Buat file controllers/controllers.py untuk membuat endpoint JSON.
5. **Buat Template:** Buat file XML yang berisi template QWeb untuk halaman graph.
6. **Tambahkan Asset:** Edit \_\_manifest\_\_.py untuk memuat library JS (seperti D3.js) dan file JS kustom Anda.
7. **Install Modul:** Update aplikasi di Odoo dan install modul batak\_community.

**Kesimpulan dan Saran**

* **Start Small:** Fokus dulu pada penginputan data yang benar dengan field father\_id. Tanpa data hubungan ini, tarombo tidak bisa dibangun.
* **Gunakan Leluhur Umum:** Untuk mulai membuat tree, tentukan satu leluhur bersama yang diketahui oleh semua anggota marga. Input data dimulai dari dia dan terus ke bawah.
* **Visualisasi adalah Tantangan:** Membangun visualisasi tree yang baik membutuhkan usaha di JavaScript. Anda bisa mencari contoh "D3.js collapsible tree" dan mengadaptasinya untuk Odoo.

Dengan mendesain struktur database ini, Anda telah meletakkan pondasi yang sangat kuat. Selanjutnya, tinggal mengembangkan front-end untuk visualisasi yang memukau. **Horas!**