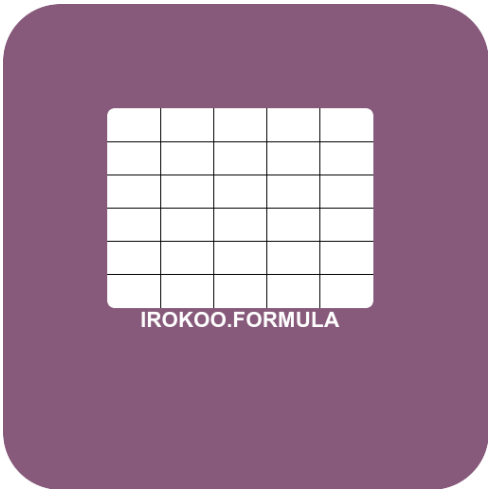

Documentation Formules Spreadsheet Evolution

Sylvain Boutet

18 mai 2025

Odoo Version 18.0

Table des matières

1 Introduction **3**

2 Liste des formules disponibles **3**

2.1 IROKOO.GET_FIELD . . . 3

2.1.1 Syntaxe . . . 3

2.1.2 Paramètres . . . 3

2.1.3 Retourne . . . 4

2.1.4 Exemple . . . 4

2.1.5 Exemple avec champ relationnel . . . 4

2.2 IROKOO.GET_IDS . . . 4

2.2.1 Syntaxe . . . 4

2.2.2 Paramètres . . . 4

2.2.3 Formats de filtre supportés . . . 5

2.2.4 Retourne . . . 5

2.2.5 Exemple . . . 5

2.3 IROKOO.GET_SUM . . . 5

2.3.1 Syntaxe . . . 5

2.3.2 Paramètres . . . 5

2.3.3 Retourne . . . 6

2.3.4 Exemple . . . 6

2.4 IROKOO.GET_GROUPED_IDS . . . 6

2.4.1 Syntaxe . . . 6

2.4.2 Paramètres . . . 6

2.4.3 Retourne . . . 6

2.4.4 Exemple . . . 7

2.5 IROKOO.SUM_BY_DOMAIN . . . 7

2.5.1 Syntaxe . . . 7

2.5.2 Paramètres . . . 7

2.5.3 Retourne . . . 7

2.5.4 Exemple . . . 7

2.6 IROKOO.COUNT_BY_DOMAIN . . . 8

2.6.1 Syntaxe . . . 8

2.6.2 Paramètres . . . 8

2.6.3 Retourne . . . 8

2.6.4 Exemple . . . 8

3 Utilisation avancée **9**

3.1 Combinaison de formules . . . 9

3.2 Filtres avec l'opérateur IN . . . 9

3.3 Système de mise en cache . . . 9

3.4 Conseils pour les performances . . . 9

3.5 Utilisation de cellules de référence comme filtres . . . 10

3.6 Gestion des dates . . . 10

3.7 Conseils pour l'optimisation . . . 10

4 Dépannage **11**
4.1 Erreurs courantes . . . 11
4.2 Problèmes de chargement . . . 11
4.3 Problèmes d'accès . . . 11

5 Aspects techniques et défis rencontrés **12**
5.1 Chargement asynchrone des données . . . 12
5.2 Simulation de GET_FIELD pour résoudre les blocages . . . 12
5.3 Système de cache intelligent . . . 12
5.4 Gestion des erreurs et résilience . . . 12
5.5 Optimisation de COUNT_BY_DOMAIN . . . 13
5.6 Conseils de développement pour l'extension . . . 13

6 Exemple de tableau de bord **13**
6.1 Objectif du tableau de bord . . . 13
6.2 Architecture à deux feuilles . . . 14
6.3 Implémentation étape par étape . . . 14
6.3.1 Étape 1 : Création et paramétrage des feuilles . . . 14
6.3.2 Étape 2 : Préparation des données de référence dans l'onglet principal 14
6.3.3 Étape 3 : Préparation des requêtes complexes dans la feuille Data . 15
6.3.4 Étape 4 : Configuration des colonnes de dates et périodes . . . 15
6.3.5 Étape 5 : Évolution mensuelle des ventes . . . 15
6.3.6 Étape 6 : Configuration des Top 5 clients . . . 16
6.3.7 Étape 7 : Configuration des Top 5 produits . . . 16
6.3.8 Étape 8 : Taux de conversion des devis . . . 17
6.3.9 Étape 9 : Répartition par équipe de vente . . . 17
6.3.10 Étape 10 : Ajout de graphiques . . . 17
6.4 Astuces pour optimiser le tableau de bord . . . 17

1 Introduction

Le module **Spreadsheet Evolution** étend les fonctionnalités des feuilles de calcul Odoo en ajoutant des formules avancées permettant d'exploiter pleinement les données du système directement dans vos spreadsheets sans export préalable.

Ces formules permettent :

— L'accès direct aux champs de n'importe quel enregistrement dans Odoo

— La recherche conditionnelle d'enregistrements avec des filtres avancés

— L'agrégation de données (sommes, moyennes, etc.)

— Les regroupements sophistiqués

— Le calcul direct de sommes avec des domaines de filtrage

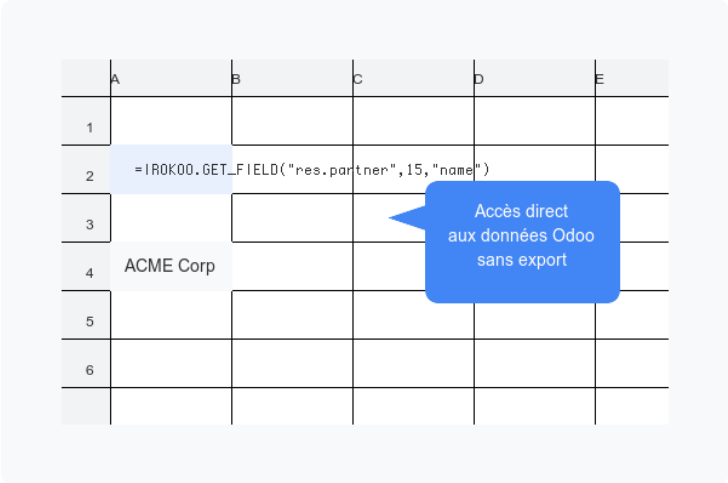

FIGURE 1 – Exemple d'utilisation des formules dans une feuille de calcul

2 Liste des formules disponibles

2.1 IROKOO.GET_FIELD

Description

Récupère la valeur d'un champ spécifique pour un enregistrement donné dans n'importe quel modèle Odoo.

2.1.1 Syntaxe

```
=IROKOO.GET_FIELD(model, id, field)
```

2.1.2 Paramètres

— **model** (chaîne) : Nom technique du modèle (ex : "res.partner")

— **id** (nombre) : ID de l'enregistrement

— **field** (chaîne) : Nom technique du champ (ex : "name", "email", "phone")

2.1.3 Retourne

La valeur du champ demandé.

2.1.4 Exemple

```
=IROKOO.GET_FIELD("res.partner", 1, "name")
```

Retourne le nom du partenaire ayant l'ID 1.

2.1.5 Exemple avec champ relationnel

```
=IROKOO.GET_FIELD("sale.order", 123, "partner_id.name")
```

Retourne directement le nom du client associé à la commande de vente ayant l'ID 123, en accédant au champ `name` à travers la relation `partner_id`.

Astuce

Cette formule fonctionne avec tous les types de champs, y compris les champs relationnels. Pour les champs relationnels, le contenu retourné sera automatiquement formaté de manière appropriée.

2.2 IROKOO.GET_IDS

Description

Récupère les IDs des enregistrements d'un modèle qui correspondent aux critères de filtre spécifiés.

2.2.1 Syntaxe

```
=IROKOO.GET_IDS(model, order, direction, limit, filters)
```

2.2.2 Paramètres

— **model** (chaîne) : Nom technique du modèle (ex : "sale.order")

— **order** (chaîne) : Champ pour le tri (ex : "date_order")

— **direction** (chaîne) : Direction du tri ("asc" ou "desc")

— **limit** (nombre) : Nombre maximum d'enregistrements à retourner (0 pour pas de limite)

— **filters** (chaîne) : Filtres séparés par des points-virgules

2.2.3 Formats de filtre supportés

— **Format explicite** : `field:operator:value`
  — Exemple : `state:in:draft,sent`
  — Opérateurs : `=`, `!=`, `>`, `<`, `>=`, `<=`, `in`, `not in`, `ilike`, `like`

— **Format simple** : `field=value`, `field>value`, etc.
  — Exemple : `amount_total>1000`

— **Format recherche texte** : `field value` pour recherches insensibles à la casse
  — Exemple : `name Comptabilité`

— **Format avec %** : `field=%valeur%` pour recherches partielles
  — Exemple : `name=%Dupont%`

2.2.4 Retourne

Une chaîne contenant les IDs séparés par des virgules.

2.2.5 Exemple

```
=IROKOO.GET_IDS("sale.order", "date_order", "desc", 10, "state:in:draft,
    sent;date_order>2023-01-01")
```

Retourne les 10 derniers devis/commandes en état brouillon ou envoyé créés après le 1er janvier 2023.

Attention !

Pour les filtres contenant des caractères spéciaux, assurez-vous d'utiliser le format explicite avec des deux-points comme séparateurs.

2.3 IROKOO.GET_SUM

Description

Calcule la somme des valeurs d'un champ pour une liste d'IDs.

2.3.1 Syntaxe

```
=IROKOO.GET_SUM(model, field, ids)
```

2.3.2 Paramètres

— **model** (chaîne) : Nom technique du modèle (ex : "sale.order")

— **field** (chaîne) : Nom technique du champ à sommer (ex : "amount_untaxed")

— **ids** (chaîne) : Liste d'IDs séparés par des virgules (généralement le résultat de GET_IDS)

2.3.3 Retourne

La somme des valeurs du champ pour les enregistrements spécifiés.

2.3.4 Exemple

```
=IROKOO.GET_SUM("sale.order", "amount_untaxed", IROKOO.GET_IDS("sale.
    order", "date_order", "desc", 0, "state=sale"))
```

Calcule la somme des montants HT de toutes les commandes confirmées.

Note

Cette formule est particulièrement utile combinée avec GET_IDS pour créer des rapports dynamiques.

2.4 IROKOO.GET_GROUPED_IDS

Description

Regroupe les enregistrements selon un champ et calcule une valeur agrégée pour chaque groupe.

2.4.1 Syntaxe

```
=IROKOO.GET_GROUPED_IDS(model, group_by, aggregate_field,
    aggregate_function, filters, limit)
```

2.4.2 Paramètres

- **model** (chaîne) : Nom technique du modèle
- **group_by** (chaîne) : Champ utilisé pour le regroupement
- **aggregate_field** (chaîne) : Champ à agréger
- **aggregate_function** (chaîne) : Fonction d'agrégation : "sum", "avg", "count", "min", "max"
- **filters** (chaîne) : Filtres (même format que GET_IDS)
- **limit** (nombre) : Nombre maximum de groupes à retourner

2.4.3 Retourne

Une chaîne contenant les valeurs du champ de regroupement, séparées par des virgules, pour les groupes classés par valeur agrégée décroissante.

2.4.4 Exemple

```
=IROKOO.GET_GROUPED_IDS("sale.order", "partner_id", "amount_total", "sum
    ", "state=sale", 5)
```

Retourne les IDs des 5 clients ayant généré le plus de chiffre d'affaires à travers les commandes confirmées.

Astuce

Cette formule est idéale pour créer des analyses de type "Top N" ou pour identifier des tendances comme les clients les plus actifs, les produits les plus vendus, etc.

2.5 IROKOO.SUM_BY_DOMAIN

Description

Calcule directement la somme d'un champ pour les enregistrements correspondant à un domaine, sans nécessiter d'étape intermédiaire.

2.5.1 Syntaxe

```
=IROKOO.SUM_BY_DOMAIN(model, field, filters)
```

2.5.2 Paramètres

— **model** (chaîne) : Nom technique du modèle

— **field** (chaîne) : Champ à sommer

— **filters** (chaîne) : Filtres (même format que GET_IDS)

2.5.3 Retourne

La somme des valeurs du champ pour les enregistrements correspondant aux filtres.

2.5.4 Exemple

```
=IROKOO.SUM_BY_DOMAIN("account.move.line", "balance", "account_id
    =401100;date>=2023-01-01;date<=2023-12-31")
```

Calcule le solde total des écritures comptables sur le compte fournisseurs 401100 pour l'année 2023.

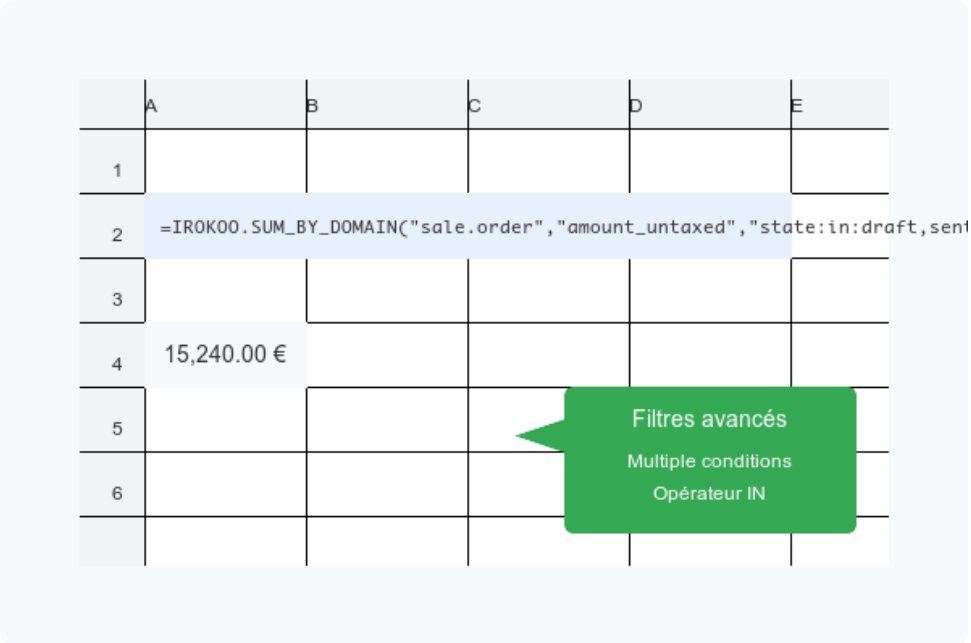

FIGURE 2 – Exemple d'utilisation de SUM_BY_DOMAIN avec filtres complexes

2.6 IROKOO.COUNT_BY_DOMAIN

Description

Compte le nombre d'enregistrements correspondant à un domaine de filtrage, sans nécessiter d'étape intermédiaire.

2.6.1 Syntaxe

```
=IROKOO.COUNT_BY_DOMAIN(model, filters)
```

2.6.2 Paramètres

— **model** (chaîne) : Nom technique du modèle

— **filters** (chaîne) : Filtres (même format que GET_IDS)

2.6.3 Retourne

Le nombre d'enregistrements correspondant aux filtres.

2.6.4 Exemple

```
=IROKOO.COUNT_BY_DOMAIN("sale.order", "state=sale;date_order
    >=2023-01-01;date_order<=2023-12-31")
```

Compte le nombre de commandes confirmées pour l'année 2023.

Astuce

Utilisez cette formule à la place de `SUM_BY_DOMAIN` avec `"1"` comme champ pour obtenir un comptage plus précis et plus performant des enregistrements.

3 Utilisation avancée

3.1 Combinaison de formules

Les formules peuvent être combinées pour créer des analyses complexes :

```
=IROKOO.GET_SUM("sale.order", "amount_untaxed",
  IROKOO.GET_IDS("sale.order", "date_order", "desc", 0,
    "state=sale;partner_id=" & IROKOO.GET_GROUPED_IDS("sale.order", "
   partner_id", "amount_total", "sum", "state=sale", 1)
  )
)
```

Cette formule calcule le montant total des ventes pour le meilleur client.

3.2 Filtres avec l'opérateur IN

Utilisez l'opérateur IN pour filtrer sur plusieurs valeurs :

```
=IROKOO.SUM_BY_DOMAIN("sale.order", "amount_total", "state:in:draft,sent
   ,sale;date>=2023-01-01")
```

3.3 Système de mise en cache

Pour améliorer les performances, les formules implémentent désormais un système de mise en cache automatique :

— Les résultats des requêtes sont stockés en mémoire pour une réutilisation rapide

— Le cache est automatiquement invalidé lors des changements de données

— Cette optimisation améliore considérablement la réactivité des tableaux de bord complexes

Astuce

Grâce au système de cache, l'actualisation des données est beaucoup plus rapide lors de modifications mineures des filtres ou paramètres.

3.4 Conseils pour les performances

— Limiter le nombre d'appels de formules en utilisant SUM_BY_DOMAIN directement quand possible

— Utiliser des limites appropriées dans GET_IDS pour réduire le volume de données

— Réutiliser les résultats de GET_IDS dans plusieurs cellules plutôt que de répéter l'appel

— Préférer le format explicite field :operator :value pour les filtres complexes

3.5 Utilisation de cellules de référence comme filtres

Il est important de noter que les formules IROKOO.* ne sont pas directement compatibles avec les filtres globaux natifs d'Odoo Spreadsheet. Cependant, vous pouvez créer un système de filtrage efficace en utilisant des cellules de référence :

— **Pour l'année** : Créez une cellule avec validation de données (liste déroulante) contenant les années disponibles, et référencez cette cellule dans vos formules

— **Pour la période** : Utilisez des cellules pour définir les dates de début et de fin, qui serviront de paramètres à toutes vos formules

— **Pour d'autres filtres** : Créez des cellules contenant les IDs des équipes, produits, etc., que vous souhaitez filtrer

Exemple de mise en place :

```
D1: "Filtre par équipe"
D2: [liste déroulante avec IDs des équipes]
```

Puis dans vos formules, référencez cette cellule :

```
=IROKOO.SUM_BY_DOMAIN("sale.order", "amount_untaxed", "state=sale;
   team_id=" & D2 & ";date_order>=" & TEXT(B2,"yyyy-mm-dd"))
```

Cette approche permet de mettre à jour dynamiquement toutes vos formules lorsque l'utilisateur change la valeur d'une cellule de filtrage.

3.6 Gestion des dates

Les formules ont été optimisées pour une meilleure gestion des filtres de date :

— Les dates sont correctement formatées quel que soit le format de la cellule source

— Les formules attendent patiemment le chargement complet des données avant d'afficher les résultats

— Les textes de chargement sont localisés en français ("Chargement des données en cours...")

Pour une utilisation optimale avec des filtres de date :

```
=IROKOO.SUM_BY_DOMAIN("sale.order", "amount_total", "date_order>=" &
   TEXT(B2,"yyyy-mm-dd") & ";date_order<=" & TEXT(B3,"yyyy-mm-dd"))
```

3.7 Conseils pour l'optimisation

— Utilisez des cellules de référence pour les paramètres communs (dates, filtres)

— Minimisez le nombre d'appels à IROKOO.GET_IDS en stockant les résultats dans des cellules intermédiaires

— Privilégiez IROKOO.SUM_BY_DOMAIN pour les calculs directs plutôt que des formules imbriquées complexes

— Pour les tableaux très volumineux, segmentez vos requêtes par période pour améliorer les performances

Ce tableau de bord vous donne un point de départ pour créer vos propres analyses adaptées à vos besoins spécifiques. Les formules peuvent être modifiées et combinées pour répondre à différentes questions métier.

4 Dépannage

4.1 Erreurs courantes

— **"Tous les paramètres sont requis"** : Vérifiez que tous les arguments obligatoires sont fournis

— **"Chargement des données en cours..."** : Patientez, les données sont en cours de chargement

— **"Aucun résultat trouvé"** : Les filtres ne correspondent à aucun enregistrement

— **"Erreur lors de l'exécution de la recherche"** : Vérifiez la syntaxe des filtres et l'existence du modèle

4.2 Problèmes de chargement

Si vous rencontrez des problèmes avec les formules affichant "Chargement des données en cours..." trop longtemps :

— Vérifiez que les filtres de date sont correctement formatés avec `TEXT(date,"yyyy-mm-dd")`

— Réduisez la plage de données en ajoutant des filtres plus restrictifs

— Vérifiez les performances de votre serveur Odoo, des requêtes complexes peuvent prendre du temps

— Consultez les logs du navigateur pour identifier d'éventuelles erreurs JavaScript

4.3 Problèmes d'accès

— Les formules respectent les droits d'accès de l'utilisateur

— Un utilisateur ne peut accéder qu'aux données auxquelles il a droit

— Si une formule retourne une erreur pour certains utilisateurs, vérifiez leurs droits d'accès

Attention !

Si vous rencontrez des problèmes d'accès, assurez-vous que l'utilisateur a les droits nécessaires pour accéder au modèle et aux champs spécifiés dans la formule.

5 Aspects techniques et défis rencontrés

Au cours du développement de ce module, nous avons dû faire face à plusieurs défis techniques. Cette section présente les principales difficultés rencontrées et les solutions mises en œuvre pour les surmonter. Ces informations peuvent être utiles pour les développeurs souhaitant étendre les fonctionnalités ou comprendre les choix d'implémentation.

5.1 Chargement asynchrone des données

- **Problème** : Les formules s'exécutent de manière asynchrone, ce qui peut entraîner des affichages "Chargement des données en cours..." persistants, notamment avec les filtres de date.
- **Solution** : Implémentation d'un système de gestion avancé des promesses JavaScript, avec détection de l'état de chargement et retry automatique en cas d'échec.
- **Résultat** : Une expérience utilisateur plus fluide et des formules qui attendent correctement que les données soient disponibles.

5.2 Simulation de GET_FIELD pour résoudre les blocages

- **Problème** : Dans certaines situations complexes impliquant des champs relationnels et des filtres de date, les formules GET_GROUPED_IDS restaient bloquées en chargement.
- **Solution** : Nous avons implémenté une technique de "simulation" de GET_FIELD dans le code de GET_GROUPED_IDS, permettant de pré-charger les données relationnelles nécessaires avant de commencer le calcul d'agrégation.
- **Exemple** : Pour les filtres comme "order_id.date_order>=", le module pré-charge maintenant les dates avant de filtrer, évitant ainsi les blocages.

5.3 Système de cache intelligent

- **Problème** : Répétition de requêtes identiques, particulièrement sur les tableaux de bord complexes, entraînant des performances insuffisantes.
- **Solution** : Implémentation d'un système de cache à deux niveaux :
  - Cache de premier niveau pour les résultats de recherche (domaines)
  - Cache de second niveau pour les résultats d'agrégation (sommes, comptes, etc.)
- **Avantage** : Réduction drastique du nombre de requêtes, avec invalidation automatique du cache lors des modifications de données.

5.4 Gestion des erreurs et résilience

- **Problème** : Certaines formules échouaient silencieusement ou bloquaient l'interface utilisateur lors d'erreurs.
- **Solutions** :

  - Localisation française des messages d'erreur pour une meilleure compréhension
  - Implémentation d'un système de logging détaillé dans la console navigateur
  - Mécanisme de fallback permettant de retourner des résultats partiels plutôt qu'une erreur complète
- **Résultat** : Un module plus robuste qui guide l'utilisateur en cas de problème.

5.5 Optimisation de COUNT_BY_DOMAIN

- **Problème** : Utilisation inefficace de SUM_BY_DOMAIN avec "1" comme champ pour compter les enregistrements.
- **Solution** : Création d'une formule dédiée COUNT_BY_DOMAIN utilisant l'API native de comptage d'Odoo.
- **Bénéfice** : Performances améliorées et meilleure lisibilité des formules dans les tableaux de bord.

5.6 Conseils de développement pour l'extension

Pour les développeurs souhaitant étendre ce module avec de nouvelles formules, nous recommandons de :

- Suivre le pattern de gestion asynchrone mis en place pour assurer la compatibilité
- Intégrer vos formules dans le système de cache existant
- Tester intensivement avec des cas complexes impliquant des champs relationnels et des filtres de date
- Utiliser la console du navigateur pour déboguer lors du développement (les logs détaillés y sont disponibles)
- Prévoir des mécanismes de fallback pour retourner des résultats partiels plutôt que des erreurs complètes

Astuce

Pour le développement de nouvelles formules, inspectez le code source de GET_GROUPED_IDS qui illustre les meilleures pratiques de gestion asynchrone des données et d'utilisation du cache.

6 Exemple de tableau de bord

Cette section présente un exemple complet de construction d'un tableau de bord d'analyse des ventes à l'aide des formules du module Spreadsheet Evolution.

6.1 Objectif du tableau de bord

Notre tableau de bord vise à répondre aux questions suivantes :

— Quelle est l'évolution mensuelle des ventes sur l'année en cours ?

— Qui sont nos 5 meilleurs clients et leur chiffre d'affaires ?

— Quels sont nos 5 produits les plus vendus (en valeur et en quantité) ?

— Quel est le taux de conversion de nos devis en commandes ?

— Comment se répartissent nos ventes par équipe de vente ?

6.2 Architecture à deux feuilles

Une bonne pratique pour construire des tableaux de bord complexes consiste à utiliser deux feuilles distinctes :

— **Feuille principale** : "Tableau de bord Ventes" - Contient la présentation et les visualisations

— **Feuille de données** : "Data" - Stocke les requêtes complexes et les résultats intermédiaires

Cette architecture présente plusieurs avantages :

— Les requêtes complexes (notamment GET_GROUPED_IDS) sont isolées et plus faciles à déboguer

— Les formules de la feuille principale restent plus lisibles et maintenues

— Le cache fonctionne mieux car les requêtes similaires sont centralisées

— L'affichage est plus propre, la feuille de données pouvant être masquée

6.3 Implémentation étape par étape

6.3.1 Étape 1 : Création et paramétrage des feuilles

— Créez une première feuille nommée "Tableau de bord Ventes"

— Créez une seconde feuille nommée "Data"

— Dans les paramètres de la feuille "Data", vous pouvez masquer les colonnes contenant les formules complexes

6.3.2 Étape 2 : Préparation des données de référence dans l'onglet principal

Dans la feuille principale, configurez les filtres globaux :

```
C1: "Année en cours"
D1: =YEAR(TODAY())
C2: "Date début"
D2: =DATE(D1,1,1)
C3: "Date fin"
D3: =DATE(D1,12,31)
```

6.3.3 Étape 3 : Préparation des requêtes complexes dans la feuille Data

Dans la feuille "Data", préparez les requêtes de regroupement qui seront utilisées à plusieurs endroits :

```
# Top 5 clients par montant de vente
B2: =IROKOO.GET_GROUPED_IDS("sale.order", "partner_id", "amount_total",
    "sum", "state=sale;date_order>=" & TEXT('Tableau de bord Ventes'!D2,"
    yyyy-mm-dd") & ";date_order<=" & TEXT('Tableau de bord Ventes'!D3,"
    yyyy-mm-dd"), 5)

# Top 5 produits par montant de vente
B6: =IROKOO.GET_GROUPED_IDS("sale.order.line", "product_id", "
    price_subtotal", "sum", "order_id.state=sale;order_id.date_order>=" &
     TEXT('Tableau de bord Ventes'!D2,"yyyy-mm-dd") & ";order_id.
    date_order<=" & TEXT('Tableau de bord Ventes'!D3,"yyyy-mm-dd"), 5)

# Top équipes de vente
B8: =IROKOO.GET_GROUPED_IDS("sale.order", "team_id", "amount_total", "
    sum", "state=sale;date_order>=" & TEXT('Tableau de bord Ventes'!D2,"
    yyyy-mm-dd") & ";date_order<=" & TEXT('Tableau de bord Ventes'!D3,"
    yyyy-mm-dd"), 2)
```

Attention !

Il est crucial de préparer l'onglet "Data" avant de configurer les formules dans l'onglet principal. Sinon, vous obtiendrez des erreurs de référence puisque les formules de l'onglet principal feront référence à des données qui n'existent pas encore.

Astuce

Ajoutez des commentaires explicatifs dans la feuille Data à côté des formules pour identifier facilement leur fonction. Ces commentaires peuvent être mis dans la colonne A.

6.3.4 Étape 4 : Configuration des colonnes de dates et périodes

Retournez dans l'onglet principal et configurez les périodes mensuelles :

```
# Colonnes pour les dates (peuvent être cachées)
A6: =MONTH.START(B6)            # Début du mois
B6: =EOMONTH(D2,0)              # Fin du premier mois
A7: =MONTH.START(B7)            # Début du mois suivant
B7: =EOMONTH(B6,1)              # Fin du mois suivant
# Etc. pour chaque mois...
```

6.3.5 Étape 5 : Évolution mensuelle des ventes

Configurez maintenant l'affichage des données mensuelles :

```
# Noms des mois et données
C5: "Mois"
D5: "Total ventes HT"
E5: "Nbre cmde"
```

```
C6: "Janvier"
D6: =IROKOO.SUM_BY_DOMAIN("sale.order", "amount_untaxed", "state=sale;
    date_order>=" & TEXT(A6,"yyyy-mm-dd") & ";date_order<=" & TEXT(B6,"
    yyyy-mm-dd"))
E6: =IROKOO.COUNT_BY_DOMAIN("sale.order", "state=sale;date_order>=" &
    TEXT(A6,"yyyy-mm-dd") & ";date_order<=" & TEXT(B6,"yyyy-mm-dd"))

# Répéter pour tous les mois...

# Ajouter un total pour le semestre
C18: "TOTAL S1"
D18: =SUM(D6:D11)
E18: =SUM(E6:E11)
```

6.3.6 Étape 6 : Configuration des Top 5 clients

Utilisez maintenant les données regroupées de la feuille "Data" pour afficher les meilleurs clients :

```
H4: "Top 5 clients"
H5: "Nom"
I5: "Montant cde"

# Colonne pour stocker les IDs (peut être cachée)
G5: "ID"
G6: =TRANSPOSE(INDEX(SPLIT(Data!B2,","),1))

# Affichage du nom et montant pour chaque client
H6: =IFERROR(IROKOO.GET_FIELD("res.partner",G6,"name"),"")
I6: =IFERROR(IROKOO.SUM_BY_DOMAIN("sale.order", "amount_untaxed", "state
    =sale;partner_id=" & G6 & ";date_order>=" & TEXT(D$2,"yyyy-mm-dd") &
    ";date_order<=" & TEXT(D$3,"yyyy-mm-dd")),"")

# Répéter pour les lignes suivantes (G7-I10) en incrémentant les réfé
    rences
```

6.3.7 Étape 7 : Configuration des Top 5 produits

Même principe pour les produits, en utilisant le regroupement stocké dans la feuille "Data" :

```
K4: "Top 5 produits"
K5: "Produit"
L5: "Qté"
M5: "Montant cde"

# Colonne pour stocker les IDs (peut être cachée)
J5: "ID"
J6: =TRANSPOSE(INDEX(SPLIT(Data!B6,","),1))

# Affichage des informations pour chaque produit
K6: =IROKOO.GET_FIELD("product.product",J6,"name")
L6: =IROKOO.SUM_BY_DOMAIN("sale.order.line", "product_uom_qty", "
    order_id.state=sale;product_id=" & J6 & ";order_id.date_order>=" &
    TEXT(D2,"yyyy-mm-dd") & ";order_id.date_order<=" & TEXT(D3,"yyyy-mm-
    dd"))
```

```
M6: =IROKOO.SUM_BY_DOMAIN("sale.order.line", "price_subtotal", "order_id
    .state=sale;product_id=" & J6 & ";order_id.date_order>=" & TEXT(D2,"
    yyyy-mm-dd") & ";order_id.date_order<=" & TEXT(D3,"yyyy-mm-dd"))

# Répéter pour les lignes suivantes
```

6.3.8 Étape 8 : Taux de conversion des devis

Calculons le taux de conversion des devis en commandes :

```
C24: "Taux de conversion des devis"
A25: "Nombre total de devis"
D25: =IROKOO.COUNT_BY_DOMAIN("sale.order", "date_order>=" & TEXT(D2,"
    yyyy-mm-dd") & ";date_order<=" & TEXT(D3,"yyyy-mm-dd"))
A26: "Nombre de commandes confirmées"
D26: =IROKOO.COUNT_BY_DOMAIN("sale.order", "state=sale;date_order>=" &
    TEXT(D2,"yyyy-mm-dd") & ";date_order<=" & TEXT(D3,"yyyy-mm-dd"))
A27: "Taux de conversion"
D27: =IF(D25=0,0,D26/D25)
```

6.3.9 Étape 9 : Répartition par équipe de vente

Analysons les performances par équipe de vente en utilisant la requête stockée dans la feuille "Data" :

```
# Colonne cachée pour les IDs d'équipe
G24: "ID"
G25: =TRANSPOSE(INDEX(SPLIT(Data!B8,","),1))

H24: "Répartition par équipe de vente"
H25: =IROKOO.GET_FIELD("crm.team",G25,"name")
I25: =IROKOO.SUM_BY_DOMAIN("sale.order", "amount_untaxed", "state=sale;
    team_id=" & G25 & ";date_order>=" & TEXT(D2,"yyyy-mm-dd") & ";
    date_order<=" & TEXT(D3,"yyyy-mm-dd"))
```

6.3.10 Étape 10 : Ajout de graphiques

Pour finaliser le tableau de bord, ajoutez deux graphiques principaux :

— Un graphique en barres montrant l'évolution mensuelle des ventes (utilisant la plage D6 :D17 pour les données et C6 :C17 pour les étiquettes)

— Un graphique en secteurs montrant la répartition par clients (utilisant la plage I6 :I17 pour les données et H6 :H17 pour les étiquettes)

6.4 Astuces pour optimiser le tableau de bord

— **Masquer les colonnes techniques** : Les colonnes contenant uniquement des IDs ou des calculs intermédiaires peuvent être masquées pour améliorer la lisibilité

— **Utiliser IFERROR** : Cette fonction permet de gérer élégamment les cas où des données seraient absentes

— **Centraliser les requêtes complexes** : Placer les formules GET_GROUPED_IDS dans la feuille "Data" améliore les performances et facilite la maintenance

— **Référencer la feuille principale pour les filtres** : Utiliser 'Tableau de bord Ventes'!D2 comme référence garantit que toutes les requêtes utilisent les mêmes paramètres

— **Utiliser les fonctions TRANSPOSE et INDEX avec SPLIT** : Cette combinaison permet d'extraire facilement les IDs individuels de résultats de regroupement

Astuce

Pour extraire un élément spécifique d'un résultat de type liste (comme celui retourné par GET_GROUPED_IDS), utilisez la combinaison TRANSPOSE(INDEX(SPLIT(résultat,","),index_voulu)) où index_voulu est le numéro de l'élément à extraire (1 pour le premier).

Ce tableau de bord vous donne un point de départ solide pour créer vos propres analyses adaptées à vos besoins spécifiques. Les techniques utilisées ici peuvent être facilement adaptées pour suivre d'autres indicateurs clés ou pour créer des tableaux de bord pour d'autres modules d'Odoo.