Cul1

了解。**“ドメイン完全非依存”で動くためのカラム（スキーマ）設計**を、限界まで具体化してまとめます。
ここに書いた仕様だけ満たせば、医療・教育・小売・金融・政策・ネットワーク等あらゆる用途を**追加設定なし**で処理できます。
（注：以下は**保存/検証/自動推定/可視化**のすべてに使う“基礎仕様”。UI のラベルや図はドメインに合わせて自動で言い換えますが、**このスキーマ自体は不変**です。）

---

1. スキーマの思想（Role = 役割、Column = 生列）

- **列名は自由**（日本語・英語・略語・蛇腹OK）。
- システムは「列名辞書＋統計的検出」で**各列の“役割（Role）”を自動割当**します。
- 推定器と品質ゲートは**Role**だけを見ます（中身の値・型・単位は標準化して渡す）。
- **複数候補が競合**するときは**信頼度スコア**で勝者を選び、UI で人手修正も可能（履歴と監査を保存）。

---

2. 正準ロール一覧（Canonical Roles）

> *これが**最小完結**の役割集合。列は 0–N 本でもよい（ない場合は推定器が自動フォールバック）。*

ロール名	必須/任意	期待型	説明（非依存）	代替/別表現
`unit_id`	準必須	string/int	個体キー（人/企業/デバイス/店舗/ノード）	複合キーは `unit_id` に連結
`time`	任意（強推奨）	datetime/period/int	観測時点（連続/離散）	`date`, `timestamp`, 週番号など
`w`	準必須	binary/multi/real	介入/施策/処置（0/1, A/B/C, 用量）	`treatment`, `variant`, `dose`
`y`	準必須	numeric	アウトカム（利益/点数/生存/損失…）	比率/金額/スコアも可
`x_*`	任意	mix	共変量（特徴量、交絡・予測用）	列数制限なし
`cost`	任意	numeric	施策コスト（単発/単位/固定）	`price`, `spend`
`weight`	任意	positive numeric	標本/集計ウェイト	設けなければ 1
`ps`/`logit_ps`	任意	[0,1]/real	傾向スコア or そのロジット	未提供なら学習
`cluster_id`	任意	string/int	クラスタ/グループ（クラス/病棟/地域）	
`network_edge`	任意	table	エッジ表（u,v,weight,type）	別ファイル許容
`instrument`	任意	mix	操作変数（IV/RD 境界/距離/割当）	
`policy_boundary`	任意	mix	閾値/地理境界（RD/政策）	
`exposure`	任意	[0,1]/real	近傍曝露度（ネットワーク）	自動生成も可
`y_aux_*`	任意	mix	副次アウトカム（副作用/リスク/履歴）	
`domain_hint`	任意	string[]	自動判定した参考タグ	UI での表示専用

> ***準必須：`unit_id` / `w` / `y`。***
> ***`time` は時系列因果や DID/TVCE を使うならほぼ必須。***
> *以外は未提供でも**自動学習やフォールバック**で動く（精度・CASに影響）。*

---

3. 型・単位・表現の許容範囲（標準化規約）

- **整数/小数/ブーリアン/カテゴリ/日付**は自動判定。
- **カテゴリ**はレベル数が多いと自動で One-Hot/Target Encoding を選択（リーキー対策込み）。
- **通貨/割合/ログ**は辞書＋値域で判定し、**内部は SI/ISO 正規化**
  - 通貨：`JPY`, `USD` 等 → **円/ドルに正規化**（UI では元通貨表示可能）
  - 割合：`0–1` か `%` を認識 → 内部は `0–1`
  - ログ列：`log_*`/`ln_*` は自動で逆変換情報を保持（CI 変換に使用）
- **日時**：ISO/UNIX/Excel 序列を受容。タイムゾーンは `Asia/Tokyo` を既定に補完。

---

4. JSON スキーマ（機械可読）

```
{
  "$id": "cqox.schema.v1",
  "type": "object",
  "properties": {
    "unit_id": { "type": ["string","integer"] },
    "time":    { "type": ["string","integer","number"], "description": "ISO8601 / epoch / period index" },
    "w":       { "oneOf": [
                  { "type":"integer", "enum": [0,1] },
                  { "type":"string" },          // multi-arm A/B/C
                  { "type":"number" }           // dose
                ]},
    "y":       { "type":["number","integer"] },
    "x":       { "type":"object", "additionalProperties": true },
    "cost":    { "type":["number","integer"] },
    "weight":  { "type":["number","integer"], "minimum": 0 },
    "ps":      { "type":"number", "minimum": 0, "maximum": 1 },
    "logit_ps":{ "type":"number" },
    "cluster_id": { "type": ["string","integer"] },
    "instrument": { "type": ["number","integer","string"] },
    "policy_boundary": { "type": ["number","integer","string"] },
    "exposure": { "type": ["number"], "description":"0..1 or real intensity" },
    "y_aux":   { "type":"object", "additionalProperties": true }
  },
  "required": ["unit_id","w","y"],
  "additionalProperties": true
}
```

> *CSV の場合は列がフラット。`x_*` / `y_aux_*` の接頭辞で解釈します。*

---

5. 代表的な最小 CSV と 推奨 CSV

5.1 最小（パネルでない一括観測）

```
unit_id,w,y
u001,0,120.5
u002,1,150.0
...
```

5.2 推奨（時系列 + 共変量）

```
unit_id,date,w,y,price,margin,cohort,segment,weight,cluster_id
u001,2025-09-01,0,120.5,1000,0.3,2024Q3,new,1,store-17
u001,2025-09-02,1,150.0, 980,0.3,2024Q3,new,1,store-17
...
```

5.3 ネットワーク同梱（別表）

- メイン CSV：`unit_id,...`
- エッジ CSV：`src,dst,weight,type`（任意、無向/有向どちらも可）
- `network_edge` は API で**別ファイル**アップロードも可（`dataset_id_edge` で紐付け）。

---

6. 多値処置（multi-arm）・**連続処置（dose）**の扱い

- `w` が **文字列カテゴリ**（例：A/B/C）：
  - 自動で **one-vs-rest** と **pairwise** の両方を作成し、**複数推定器**に渡す。
  - 「基準群」は自動（頻度最大 or コスト最小）／UI で上書き可能。
- `w` が **実数（dose）**：
  - **Dose-Response** 推定を有効化（単調性ゲートと連動）。
  - **離散化カーブ**と**連続推定（Spline/GP）** を両方出力。

7. アウトカム y の型と変換

- **連続**（金額/点数）：標準
- **二値**（0/1：成功/失敗）：ロジットリンクを併用、**CI は比例スケール**で返す
- **カウント**（非負整数）：Poisson/NegBin を内部参照、過分散を自動検出
- **時間到達**（生存）：`y` にイベント指標、`y_aux_time` に到達時間を置けば**擬似KM**可
- 変換：`log(y+ε)` を内部で持つことあり（CI 変換は自然尺度に戻して出力）

8. 共変量 x_* の指針

- 名前に制約なし（`x_age`, `x_income`, `x_grade`…不要。任意列は**すべて x** に流し込む）。
- **リーキー検出**：`time` 後にしか分からない列（未来情報）を自動フラグ→**除外**推奨。
- **高相関/多重共線**：VIF>10 を警告。必要に応じて Drop/正則化に回す。
- **欠損**：MCAR/MAR を簡易判定し、IPW/多重代入の選択肢を提示。

9. Propensity（`ps` / `logit_ps`）

- 与えれば**尊重**、与えないなら**自動学習**（GBDT/ロジット）。
- 片側が極端（0/1 近傍）は**Overlap/Tail Gate**で赤警告。
- 自動保存：`ps_model_hash` をプロベナンスに記録。

10. コスト・収益・重み

- `cost`：施策コスト（1観測あたり or 固定）。未提供なら 0。
- 収益計算は内部で `incremental_profit = ATE × scale − cost` を生成（UI で式表示）。
- `weight`：標本ウェイト/ポストストラティフィケーション重みを許容。

11. クラスタ・ネットワーク・IV/RD

- `cluster_id`：層化/混合効果/マルチレベルに使用。
- `network_edge`：曝露 `exposure` を派生（k-hop比率、距離減衰）。`exposure` を直接持っていてもOK。
- `instrument` / `policy_boundary`：IV/RD 推定に利用。**ない場合は自動探索**（例：曜日・距離・在庫）も試みる。

12. 検証パイプライン（入力 → 標準化 → 役割決定 → バリデーション）

1. **型推定**：各列を int/float/bool/cat/datetime に推定
2. **単位/スケール推定**：通貨/割合/ログ…を辞書＋値域で判定
3. **候補役割の抽出**：辞書一致 ＋ 正規表現 ＋ 統計特徴（分布, 自己相関, レベル数）
4. **スコアリング**：`S = α*name_match + β*type_match + γ*stat_signature + δ*coherence`
5. **勝者選出**：Role ごとに top-1（閾値未満なら「未確定」）
6. **クロス整合**：`unit_id` の一意性、`time` の単調性、`w` の値域、`y` の有界性など
7. **フォールバック**：`ps` 不在→学習、`time` 不在→DID/TVCE を無効化、`network_edge` 不在→通常ATE
8. **UI へ提示**：候補と信頼度、未確定ロールの**TODO**、推奨変換（ログ/トリミング/再スケール）
9. **監査記録**：確定マッピング、候補、スコア、辞書バージョン、検証ログを保存

主な検証ルール例

- `unit_id`×`time` の重複行が多い → 集計orキー見直し警告
- `w` が 0/1 以外を含むのに dose として一様に連続 → multi-arm/dose **曖昧フラグ**
- `y` が負値不可のはず（比率/カウント）で負値あり → 異常値フラグ
- `cluster_id` のユニーク数 = 行数 → **実質ID**（クラスタとして不適）警告
- `ps` が [0,1] 外 → スケール誤り（%か小数か）補正提案

13. 自動ロール推定（擬似コード）

```
def infer_roles(df, dictionaries):
    cand = defaultdict(list)
    for col in df.columns:
        t = infer_type(df[col])              # int/float/bool/cat/datetime
        u = infer_unit(df[col], dictionaries["units"])
        nm_score = name_match(col, dictionaries["columns"])  # tokenベース
        stat = stat_signature(df[col], t)    # 値域/尖度/レベル数/時系列性 etc.
        for role in ALL_ROLES:
            s = (
                0.50*nm_score.get(role,0) +
                0.25*type_compat(t, role) +
                0.15*stat_compat(stat, role) +
                0.10*coherence_with_others(df, role)
            )
            cand[role].append((col, s))
    mapping = {}
    conf = {}
    for role, pairs in cand.items():
        col, s = max(pairs, key=lambda x:x[1])
        if s >= THRESHOLD[role]:
            mapping[role] = col; conf[role] = s
    # 交差制約の修正（例：unit_idのユニーク率が低い→次点へ）
    mapping, conf = post_adjust(df, mapping, conf)
    return mapping, conf, candidate_lists(cand)
```

スコア閾値の典型

- **`unit_id`**: 0.70 / **`w`**: 0.65 / **`y`**: 0.65 / **`time`**: 0.60 / 他: 0.55
  閾値未満は「未確定」にして UI で修正依頼。

---

14. 曖昧さ・競合の解決規則

- **勝者1つ＋次点候補**を UI に提示（信頼度バー）。
- **`w`**: dose / multi-arm の両解があり得る場合、**データ分布**で判定（ユニーク値の個数・順序性・間隔の規則性）。
- **`time`**: 序列性とレンジで「日付 vs 期間番号」を判別。
- ルールで解けないときは**明示入力を求める**（最小5クリック）。

---

15. 監査・再現（プロベナンス）

- **確定マッピング**（誰が/いつ/どう修正）
- **候補とスコア**（自動判定の根拠）
- **辞書バージョン**（columns.json/units.json のハッシュ）
- **変換ログ**（スケール変換/トリム/外れ処理）
- **乱数シード/コードハッシュ**（推定器）
  → すべてジョブごとに JSON へ保存し、レポートに添付。

---

16. エラーカタログ（代表）

- **`E-ROLE-001`**: 必須ロール欠落（unit_id / w / y）
- **`E-ROLE-002`**: unit_id が一意でない（重複率>5%）
- **`E-ROLE-003`**: w の型不正（0/1 でもカテゴリでも dose でもない）
- **`E-ROLE-004`**: y が非数/全欠損
- **`E-ROLE-005`**: time が混在フォーマット（ISO+Excel序列）
- **`E-ROLE-006`**: ps の範囲外
- **`E-ROLE-007`**: network_edge が壊れている（自己ループ/負重みのみ 等）

→ UI では**修正方針の選択肢**（自動変換・列入替・無効化）を提示。

---

17. パフォーマンス・スケール要件

- 最大列数：理論上無制限（UIの実用は数千列まで推奨）

- 行数：単体サーバで**数千万行**はストリーミング前処理に切替（要 GPU/分割）
- 自動検出・検証は**サンプルダウンサンプリング**（既定 1e6 行）で O(列数×log行数)
- メモリ：列型推定はチャンク読み、巨大 CSV は Arrow/Parquet を推奨

---

18. 典型的な「ズレ」を吸収する辞書トリック（抜粋）

- `treat`, `treatment`, `variant`, `arm`, `policy`, `exposure`, `dose` → **w**
- `revenue`, `sales`, `y`, `target`, `label`, `score`, `outcome`, `profit`, `loss` → **y**
- `id`, `user_id`, `patient_id`, `account`, `device`, `store`, `node`, `region_code` → **unit_id**
- `date`, `day`, `week`, `month`, `timestamp`, `period`, `t` → **time**
- `ps`, `p_score`, `propensity`, `logit_p`, `lp` → **ps/logit_ps**
- `group`, `class`, `cluster`, `ward`, `store_id`, `team` → **cluster_id**
- `cost`, `spend`, `price`, `cogs`, `media_cost` → **cost**

（日本語別名も同梱：`処置`, `施策`, `群`, `用量`, `売上`, `利益`, `患者ID`, `学籍番号`, `日付`, `在庫`…）

---

19. 補助テーブル（Join 戦略）

- `unit_id` が一致すれば**左外部結合**で自動マージ（`time` があれば `unit_id+time` キー）。
- 複数テーブル（ユーザ属性・取引明細・エッジ表）は、**データセット登録時に順次追加 → 統合ビュー**を生成。
- キー不整合はヒートマップで可視化（欠損ジョイン率、孤立率）。

---

20. ストリーミング/追記の扱い

- `time` がある場合、**追記（append-only）の想定でロール再推定**は不要（監査上は差分ログを保持）。
- **遡及修正**があった列はフラグ付け（学習・推定の再現性のために古いバージョンも残す）。

まとめ（実装に渡す要点）

1. **この正準ロール**だけをエンジンに渡せばよい（列名は何でもOK）。
2. **自動ロール推定**は「辞書×統計特徴×整合性」の複合スコアで決定、**信頼度と候補**を UI に出す。
3. **検証パイプライン**で型・単位・一意性・分布・整合をチェックし、**具体的な修正手順**を提案。
4. `w` の **multi-arm/dose** は自動判定し、推定器へ**両パス**生成。
5. `time/cluster/network/IV/RD` 等は**任意**だが、あれば自動で推定器が拡張モードに入る。
6. **監査・再現**のため、マッピング・候補・辞書ハッシュ・変換ログ・シードを**必ず保存**。

この仕様で「**ドメイン完全非依存**の入力取り込み→標準化→役割確定→検証→推定」まで、**追加設定ゼロ**で回せます。
実装側（Gateway/Engine/UI）のすべてが、このロール仕様に**直結**します。