### **Требования к Информационной Системе**

В текущем разделе описывается алгоритм формирования сущности dfct\_phone, относящейся к области истории контактности клиента.

### **Бизнес-описание**

В объекте dfct\_phone хранится список клиентов физлиц с историей их контатности

### **Требования к выходным данным**

|  |  |  |
| --- | --- | --- |
| **Схема** | **Объект** | **Описание** |
| example | dfct\_phone | История контактности клиентов ФЛ |

### **Источники**

|  |  |  |  |
| --- | --- | --- | --- |
| **Схема** | **Объект** | **Описание** | **Цель использования** |
| example | counterparty | Клиенты | Фильтрация по списку и отбор атрибутов |
| example | dict\_counterparty\_type\_cd | Справочник типов клиента | Определение среди клиентов физ лиц |
| example | counterparty\_x\_uniq\_counterparty | Связь глобального и локального клиентов | Сопоставление локального и глобального идентификатора клиента |
| example | counterparty\_contact | Телефоны клиента | Определение номера телефонов и флагов для результирующей таблицы |

### **Регламент и механизм загрузки**

Тип загрузки: полный срез

Регламент загрузки: ежедневный

|  |  |  |  |
| --- | --- | --- | --- |
| **Схема** | **Объект** | **Условия отбора записей** | **Описание** |
| example | counterparty | src\_cd = 'MDMP' | отбираются клиенты из системы MDM |
| example | dict\_counterparty\_type\_cd | src\_cd in ( ‘MDMP’, ‘RTLL’, ‘RTLS’, ‘CFTB’, ‘WAYN’)  counterparty\_type\_desc = ‘физическое лицо’ | отбираются клиенты из системы MDMP, RTLL, RTLS, CFTB, WAYN  отбираются физ лица |
| example | counterparty\_x\_uniq\_counterparty | src\_cd in ( ‘MDMP’, ‘RTLL’, ‘RTLS’, ‘CFTB’, ‘WAYN’) | отбираются клиенты из системы MDMP, RTLL, RTLS, CFTB, WAYN |
| example | counterparty\_contact | src\_cd in ( ‘MDMP’, ‘RTLL’, ‘RTLS’, ‘CFTB’, ‘WAYN’) | отбираются клиенты из системы MDMP, RTLL, RTLS, CFTB, WAYN |

|  |  |
| --- | --- |
| **Название шага** | **Описание алгоритма** |
| p\_dfct\_phone\_0010 | Отбираем всю историю клиентов из **counterparty**. Соединяем с **dict\_counterparty\_type** по полю counterparty\_type\_cd.  Отсекаем клиентов, у которых источник данных не равен 'MDMP'.  Отсекаем клиентов, для которых соотвествущая запись в справочнике **dict\_counterparty\_type** имеет источник не из перечня ( 'MDMP', 'RTLL', 'RTLS', 'CFTB', 'WAYN')  Отсекаем клиентов, для которых соотвествущая запись в справочнике **dict\_counterparty\_type** содержит в поле counterparty\_type\_desc не 'физическое лицо'.  Отсекаем клиентов для которых период действия версии в counterparty и в dict\_counterparty\_type не пересекаются.  Рассчитываем атрибуты:   * effective\_from\_date * effective\_to\_date |
| p\_dfct\_phone\_0020\_1 | Для записей из **counterparty\_contact** у которых источник не равен 'MDMP' производим поиск глобального идентификатора клиента пересечением с counterparty\_x\_uniq\_counterparty .  Отбрасываем записи с неподходящим источником и неперсекающимися интервалами  Рассчитываем атрибуты:   * effective\_from\_date * effective\_to\_date |
| p\_dfct\_phone\_0020\_2 | Для записей из counterparty\_contact у которых источник равен 'MDMP', глобальный идентификатор клиента берем равный локальному идентификатору/ |
| p\_dfct\_phone\_0020 | Объединяем наборы строк, полученные в p\_dfct\_phone\_0020\_1 и p\_dfct\_phone\_0020\_2. uniq\_counterparty\_rk – получается расчетным, так как берется из разных таблиц по условию.  Рассчитываем атрибуты:   * uniq\_counterparty\_rk |
| p\_dfct\_phone\_0030 | Отбрасываем из p\_dfct\_phone\_0020 записи о клиентах, которые не физ лица, путем пересечения с p\_dfct\_phone\_0010.  \* Отбрасываем где нет пересечения интервалов |
| p\_dfct\_phone\_0040\_1 | Строим интервалы действия версии для каждого телефонного номера  Заполняем значение флага notification\_flg в зависимости от типа контакта (=’ NotificPhone’)  Заполняем значение флага atm\_flg в зависимости от типа контакта (=’ ATMPhone)  Отсекаем клиентов, для которых источник не из перечня: ('MDMP', 'RTLL', 'RTLS', 'CFTB', 'WAYN')  Рассчитываем атрибуты:   * notification\_flg * atm\_flg |
| p\_dfct\_phone\_0040\_2 | Ранжируем версии номеров по дате начала  Ранжируем версии клиентов по дате начала  \*используя row\_number, мы можем получить что с одинаковыми параметрами получать разный ранг, но иначе мы можем получить, в ситуции, когда равны даты и другие условия, дубли по майн флагам  Рассчитываем вспомогательные атрибуты:   * rang\_start\_phone\_num * rang\_start\_client\_num |
| p\_dfct\_phone\_0040\_3 | Вводим цифровые поля-оценки для параметров указанных в п.6. бизнес-требований и ранжируем в рамках каждого поля, согласно требований  Рассчитываем вспомогательные атрибуты:   * range\_trust\_num * range\_quality\_num * range\_src\_num * range\_type\_num |
| p\_dfct\_phone\_0050\_1 | Дробим на минимальные интервалы действия по каждому телефону и собираем точки начала и конца интервалов по каждому телефону |
| p\_dfct\_phone\_0050\_2 | Объединяем точки начала и конца интервала в один список |
| p\_dfct\_phone\_0050\_3 | Строим минимальные интервалы действия по каждому телефону |
| p\_dfct\_phone\_0050\_4 | Наполняем значениями минимальные интервалы действия по каждому телефону |
| p\_dfct\_phone\_0050\_5 | Собираем интервалы, по которым есть пересечения по номеру телефона и дате действия |
| p\_dfct\_phone\_0050\_6 | Заполняем атрибут duplication\_flg  Рассчитываем атрибуты:   * duplication\_flg |
| p\_dfct\_phone\_0050\_7 | Заполняем вспомогательный атрибут range\_duplication\_num  Рассчитываем вспомогательные атрибуты:   * range\_duplication\_num |
| p\_dfct\_phone\_0050\_8 | Взвешиваем каждую запись на основании п.6. бизнес-требований. Присваиваем большие веса более важным полям. В данном решении предполагаем, что количество версий (интервалов) для каждого телефонного номера не превышает 10000, что верно с большим запасом на данном наборе данных. Можно избавиться от данного требования, то тогда надо будет использовать чуть более громоздкое решение дальше. Использование признака дупликации в расчетах обоих main флагов никак не влияет на результат, поэтому не буду менять единообразную оценку.  Рассчитываем вспомогательный атрибут:   * weght\_phone\_num |
| p\_dfct\_phone\_0050\_9 | Для заполнения флаг основного клиента для данного сравниваем вес каждой записи с максимальным значением на выборке для текущего номера  Рассчитываем атрибуты:   * main\_dup\_flg |
| p\_dfct\_phone\_0060\_1 | Отбираем нужные (вспомогательная) |
| p\_dfct\_phone\_0060\_2 | Склеиваем интервалы |
| p\_dfct\_phone\_0070\_1 | Дробим на минимальные интервалы действия по каждому клиенту и собираем точки начала и конца интервалов по каждому телефону |
| p\_dfct\_phone\_0070\_2 | Объединяем точки начала и конца интервала в один список |
| p\_dfct\_phone\_0070\_3 | Строим минимальные интервалы действия по каждому клиенту |
| p\_dfct\_phone\_0070\_4 | Наполняем значениями минимальные интервалы действия по каждому телефону |
| p\_dfct\_phone\_0070\_8 | Взвешиваем каждую запись на основании п.6. бизнес-требований.  Рассчитываем вспомогательный атрибут:   * weght\_client\_num |
| p\_dfct\_phone\_0070\_9 | Для заполнения флага лучшего телефона для клиента сравниваем вес каждой записи с максимальным значением на выборке для текущего клиента  Рассчитываем атрибуты:   * main\_phone\_flg |
| p\_dfct\_phone\_0080\_1 | Отбираем нужные (вспомогательная) |
| p\_dfct\_phone\_0080\_2 | Склеиваем интервалы |
| dfct\_phone | Выбираем необходимые поля, отбрасываем вспомогательные атрибуты |

#### **Описание алгоритмов заполнения сущности**