## 

ここでは、顧客データの整形を行います。**ノック21**で読み込んだcustomerに、会員区分のclass\_masterとキャンペーン区分のcampaign\_masterを結合します。

顧客データを主にして横に結合するので、**レフトジョイン**となります。 **ジョインキー**は自分で探してみましょう。

また、ジョイン前後でデータ件数が変わらないことを確認しましょう。

customer\_join = pd.merge(customer, class\_master, on="class", how="left")
customer\_join = pd.merge(customer\_join, campaign\_master, on="campaign\_id",
how="left")

customer\_join.head()

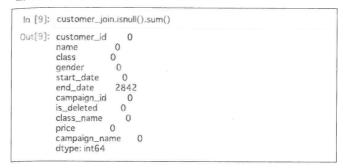
## ■図3-2:データユニオン

In [7]:	customer_join = pd.merge(customer, class_master, on="class", how="left") customer_join = pd.merge(customer_join, campaign_master, on="campaign_id", how="left") customer_join.head()											
Dut[7]:		customer_id	name	otass	gender	stort_date	end_date	campaign_ld	Is_deleted	class_name	price	campaign_name
	0	OA832399	XXXX	C01	F	2015-05-01 00:00:00	NaN	GA1	0	オールタイム	10500	30 70
	1	PL270116	XXXXX	COS	м	2015-05-01 00:00:00	NaN	GAI	0	オールライム	10500	285.79
	2	OA974876	XXXXX	G01	M	2015-05-01 00:00:00	NaN	GA1	0	オールタイム	10500	165.79
	3	HD024127	XXXXX	C01	F	2015-05-01 00:00:00	NaN	GAI	0	オールタイム	10500	385.79
	4	HD661448	XXXXX	G03	F	2015-05-01 00:00:00	NaN	CA1	0	ナイト	6000	385.70
In [8]:	<pre>print(len(customer)) print(len(customer_join))</pre>											

1行目で、会員区分のマスタデータである class\_master と、2行目で、キャンペーン区分である campaign\_master とそれぞれ結合しています。実際に先頭5行の出力結果を見ると、class\_name、price、campaign\_name 列が追加され、会員区分や金額等が分かるようにデータを整形できました。データ件数も、ジョイン前後で変化がないことが確認できます。

ジョインする際、キーが見つからないなど、上手くジョインができないと、欠 損値が自動で入ります。そのため、ジョイン後は欠損値の確認をするようにしま しょう。 customer\_join.isnull().sum()

## ■図3-3:欠損値の確認



end\_date以外は欠損値が0となっており、今回ジョインで追加した、class\_name、price、campaign\_name列にしっかりデータが入っていることが確認できました。また、end\_dateに欠損値が入っていること以外は比較的綺麗なデータであることもわかります。

また、end\_dateに欠損が入っている理由としては、退会していないユーザーは、 退会日である end\_date を保持していないため、欠損値となっていることが考え られます。

## ∅ ノック23: 顧客データの基礎集計をしよう

データ加工が完了したので、この顧客データを集計し、全体像をみていきましょう。

まずは、集計する項目を考えてみます。どの会員区分やキャンペーン区分が多いか、いつ入会/退会が多いのか、男女比率や退会するまでの期間等を集計することができることに気づくと思います。まずは、会員区分、キャンペーン区分、性別、既に退会済みかどうか(is\_deleted列)毎に全体の数を把握してみましょう。

customer\_join.groupby("class\_name").count()["customer\_id"]