Tables

- Your tables are actually your entities. When creating an entity system will create a table in the Db.
- Code First approach is used in Data Layer. https://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx

Entity

- Entity (table) designs are not approved by a Data Architect or Chief Architect. Teams are responsible for entity designs. Chief Architect can check entity designs by sampling few of the tables in DBs.
- Entities names should be in PascalCase.
- Do not use "s" suffix at the end of entity name.
- Avoid using Definition suffix for the entities. Only use if the related name is very generic like API, Action, Resource etc. Samples: ApiDefinition, ActionDefinition, ServiceDefinition, Country, City, Currency.
- Before creating your models and entities please check Data Dictionary&Db Rules.

https://docs.fimple.co.uk/userguides/en/userguide/getting-started/data-dictionary.html

- While creating models and entities, it is important to decide multi language fields from the very beginning. Multi language fields should store Json String. Sample: { "tr":"Adı", "en": "Name", "ru": "ади" } (Multi-Language-in-Backend.adoc)
- For all fields, IsRequired, MaxLength, IsUnique, etc. should be defined.

Data-Access-Layer.adoc

- Related entities (PF/FK) should be bound with required relation rule.

- Related entities (PF/FK) Considerations

    - The FK type is added to the table as "Cascade" by default. This means that when a record is deleted in the main table, the related record/s are also deleted in the detail table.
    - FK is created by EFCore as "Cascade" for navigation properties. Even if you don't define an FK relationship on your Configurator.

If a relationship is established between the definition table and the transaction table, FK Type "No Action" must be set so that the definition table is not affected when a record is deleted from the transaction table.
Code View For NoAction :

```
builder.HasOne(p => p.Parent)
                .WithMany()
                .HasForeignKey(x => x.ParentId)
                .IsRequired(false)
```

    - `.OnDelete(DeleteBehavior.Restrict);`

When a record is deleted from the table, the relevant Column is set to NULL in the Main table according to the business need.
Code View For NULL :

```
builder.HasMany(p => p.POSTransactions)
            .WithOne(y => y.Workgroup)
            .HasForeignKey(p => p.TransactionWorkgroupId)
            .IsRequired(false)
```

    - `.OnDelete(DeleteBehavior.SetNull);`

- Do not prefer ForeignKey attribute, because it set Cascade option by default for foreign keys. Use EF Core Fluent API like above.

Field Naming Standarts

- Names should be in PascalCase.
- Do not use "_" character in the name of the field.

- Do not use abbreviations in your field name. Sample: CustNo. Only well knows abbreviations may have an exception. Sample: SMSText
- Try to use already defined terms from the data dictionary for your fields. Avoid finding new names for your fields.
- Use "s" suffix for only collections. Do not use "s" suffix for any other fields. Sample: List<int> ActionIds. (For this sample there must be only on term in Data Dictionary as ActionId (int). "ActionIds" must not be included into the Data Dictionary.)
- For boolean fields, the name must start with one of these: "Can", "Is", "Has"
- Do not use table name as a prefix of the field name. Only special fields may have an exception. Sample: ModuleName is not correct in Module table, correct usage is Name.
- If a field has a lookup table then it gets an Id suffix. If the fields does not have a lookup table, no need for Id suffix. Sample: IdentificationTypeId (has lookup), CustomerRelationType (does not have look up)
- Use Name field as name info. If you need a UniqueName field, you can put it to your entity along with Name or ShortName fields. But, do not use UniqueName alone for name info.

Field Data Types

- If you use Id in the field name, the type must be int, long, guid. (There are few exceptional cases coming from outer standarts)
- If you use Name, Key, Code in the field name, the type must be string.
- For some generic names like Content, Id, Version, there can be more than one datatype. Sample: Content (string / byte[])
- For few exceptional fields, there can be two data types, one for entity, another for model. Sample: BusinessKey (string / long)

Data Types

SQL Server Database Engine type	.NET type	Notes
bit	bool	Use if the field is true or false. There there is a

		change for a third option, then use tinyiny instead.
tinyint	byte	
int	int	Don’t use smallint (Int16 / short) in .Net Code.
bigint	long	
decimal	decimal	
binary	byte[]	
date	DateTime	For only timeless data.
datetime	DateTime	UTC date should be stored for all datetime and datetimeoffset fields.
datetimeoffset	DateTimeOffset	
uniqueidentifier	Guid	
nvarchar	String / Char[]	nvarchar should be used instead of varchar. In EF Core 6, text properties are configured as Unicode by default. And so, an nvarchar fields is created in DB.
tinyint	byte	

Default and Special Fields

- When you Save a Record, below fields are being saved to DB automatically by the DBContext (The fields vary according to the types of IFiEntity).
    a. CreatedBy
    b. CreateTime
    c. UpdatedBy
    d. UpdateTime
    e. DeletedBy
    f. DeleteTime
- Every entity must have an Id field and this field must be PK.
- Entities which inherited from EntityBaseWithBaseFieldsWithIdentity has an Id column which is also PK of the table. Do not add another Id column to your entity. Wrong: FecId.

For more information about entity classes read below:
Entity-Classes.adoc

- For PK/FK relations, add virtual objects to related entities. And also detail class a foreignkey field like MainEntityNameId.

```
public virtual List<ApiParameterDefinition> Parameters { get; set; } = new List<ApiParameterDefinition>();
public virtual ApiDefinition ApiDefinition { get; set; }
```

`public virtual List<ApiParameterPropertyDefinition> Properties { get; set; } = new List<ApiParameterPropertyDefinition>(); `

- If the entity is a definition entity like Resource, Action, Event, etc. then Id column is not enough for data transport operations. So also consider to add a Code, Key or UniqueName field in your entity. Try to choose only one of Code, Key or UniqueName fields. While choosing Code, Key or UniqueName, look at entity and your concept to decide the best option.

When creating Database Model you must check this table.
If your data type in the table, you must choose available type, related to your purpose.
Check for other pre-defined data types: Fi.Persistence.Relational.Domain.DataTypes

Type Name	Purpose of using	.NET	MSSQL	EfCore Fluent API
Decimal	Use for Amount, Balance properties.	decimal	decimal(24,6)	builder.Property(m => m.Debit).IsRequired(true).HasPrecision(DataTypes.Decimal24_06)
Decimal	Use for currency fx rate properties.	decimal	decimal(18,8)	builder.Property(m => m.FxRate).IsRequired(true).HasPrecision(DataTypes.Decimal18_08)
Decimal	Use for percentage kind of properties like tax rate (%0.01) etc.	decimal	decimal(8,5)	builder.Property(m => m.TaxRate).IsRequired(true).HasPrecision(DataTypes.Decimal08_05)
Date	Use for transaction date with no time, a TransactionDate field may be added	Datetime	date	builder.Property(p => p.TransactionDate).HasColumnType(DataTypes.Date)

Sql Objects

- Use v prefix for sql views: dbo.vCustomerAddress
- If needed, use this format for stored procedures: dbo.ins_Customer, dbo.del_Customer, dbo.sel_GetCustomer, dbo.upd_Customer
- Use f prefix for sql functions: dbo.fGetCustomer
- Use tri, tru or tr prefix for sql triggers: tri_Customer, trd_Customer, tr_Customer
- Use Pascal notation for sql variables: @CustomerNo

> Try to be database agnostic and do not prefer to use triggers or any other db objects to achieve being database agnostic.