Basics

This document discusses Subtypes within the context of Dr E F Codd's *Relational Model* (not the pretenders). The methods required for a full Relational implementation (ie. to obtain the full integrity, power, and speed of the *RM*) are given, code is given where necessary.

Since there are many program suites (they are not platforms) that have "sql" in the name, and fraudulently declare compliance with the ANSI/ISO/SQL Standard, in which SQL methods are not possible, the SQL capabilities relied upon need to be asserted:

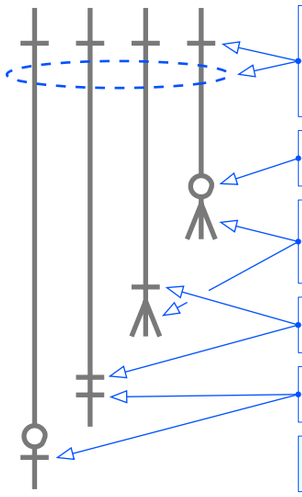
- Subtypes are supported in SQL, although without explicit verbs.
- ACID Transactions are supported in SQL, with explicit verbs.
- Constraints that call a Function are supported in SQL (indeed, a Function can be used instead of an expression, anywhere).

This document uses IDEF1X, the standard for modelling Relational databases: available since 1984; a NIST Standard since 1993.

- The original IDEF1X relation symbols and Cardinality notation was limited, and not generally recognised. The IEEE relation symbols and Cardinality notation was established later, it provides more definition and it is simpler, thus it is used instead.
- Likewise, the original IDEF1X Subtype notation permitted { Complete | Incomplete }, and did not support exclusivity. The later IEEE Subtype notation provides more definition, thus they are used instead. Given that a genuine database is structurally extended over time, { Complete | Incomplete } is irrelevant.

1.1 Implementation: Relationship

The Subtype relationship is an ordinary relationship, first, with additional properties, second. In order to understand and implement a Subtype cluster correctly, ordinary relationships must be understood first, and understood clearly. This gives the implementation as well.



Parent • Mandatory Existence
- Implemented via a `FOREIGN KEY` Constraint in the child, referencing the `PRIMARY KEY` of the parent
- In a Relational database, it is always one at the parent end, never zero-or-one, the second cross-tick can be omitted. (In Normalised data models, wherein a null parent is prohibited, both cross-ticks can be omitted !)

No Minimum or Maximum Child Count
- No control required

Maximum Child Count > 1
- For those who need it stated, if it is uncontrolled, no control required
- If the child rows need to be limited to a max, use a **`CHECK` Constraint** that calls a **simple Function**.

Minimum Child Count 1
- Implemented via a Transaction. Eg. `sp_Add_Child_tr` inserts the parent row first

Maximum Child Count 1
- Implemented via `PRIMARY KEY` in the child, identical to the parent Primary Key

Bottom Line
- If a Minimum Child Count is specified, it means the child is inserted with the parent (in a Transaction).

1.1.1 Caveat

Primary Key Demanded
- The *Relational Model* requires a `PRIMARY KEY` for each table. The Key must be *made up from the data*, ie. it must not be a surrogate.
- Other Keys that may exist are **Alternate Keys**.

Hierarchical
- All relationships in the *RM* are hierarchical. The Foreign Key in the referencing row references the Primary Key in the referenced row.
- The referenced participant must exist first, in order for the referencing participant to reference it
- Thus the participants in the relationship are never equal or "equal"

Circular Reference Prohibited
- Note that circular references are prohibited in the *Relational Model*, precisely because it is a gross Normalisation error
- Anti-relational, anti-server, RFS program suites (they cannot be called platforms) allow, and even encourage circular references.
- Thus they provide "features" purported to resolve circular references (which is fraudulent, because an error at the higher level cannot be resolved by correction at the lower level)
- Certainly, such primitive suites are attractive to the type of people who cannot Normalise data, and who are thus unaware of the gravity of nunnormalised data
- Thus "deferred constraint checking" and other "features" available in Non-SQL propagations are **not** required to support a Relational database.

Single FK Definition
- Note that precisely **one** Foreign Key reference is required.
  - *Some "theoreticians" and "professors" teach that a second Foreign Key reference, with the parent referencing the non-existent child is also "needed". The non-logic used to form this idiocy is so screwed up that it not worthy of address. It is simply incorrect.*
  - *Neither the Relational Model, nor IDEF1X, nor SQL is that stupid.*
  - *A single Foreign Key in the child, referencing the parent, establishes the relationship, and the relationship thus consists of two participants.*
  - *If Fred is Sally's father, we know from that single fact that Sally is Fred's daughter.*
- An additional, second definition for a single relationship is (a) redundant, and (b) constitutes a circular reference, which is prohibited.
- Insanities such as 'deferred constraint checking' that are required for schizophrenics, are not required for normal humans.

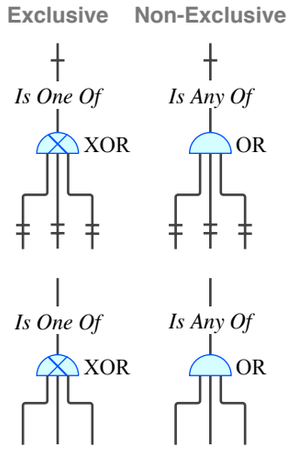
Record Filing Systems
- They are characterised by their use of surrogates, labelled as "keys". A surrogate is not made up from the data. A non-key has none of the properties of a Key,
- These have none of the Relational Integrity (as distinct from Referential Integrity), Relational power, or Relational speed that Relational Databases enjoy.
- "Candidate key" is a rebellion against the *RM*, a refusal to accept the concept of Primary Key, which is demanded. Once the election is over, there are no candidates, only one winner and a bunch of losers. It is typically used in Record Filing Systems, beloved of "theoreticians", to obtain some fraction of the integrity and power of Relational databases in their RFS.

IDEF1X Notation

1.2 Implementation: Subtype Relationship

- The relationship (across the set of Subtypes) is either Exclusive or Non-exclusive:
  - **Exclusive**: there is exactly one Basetype:Subtype pair, it is an XOR Gate.
    The Predicate is always *Is One Of*.
  - **Non-exclusive**: there is at least one Basetype:Subtype pair, it is an OR Gate.
    The Predicate is always *Is Any Of*.
- In either case, each Basetype::Subtype pair is treated as a single logical row, and should be perceived as such (yes, of course, at the DDL level only, it is two tables, but at the modelling level, it is a single logical unit):

  Person::Male is a single logical Person (of sex Male)
- The relation is formed by declaring a `FOREIGN KEY` in the Subtype (the child), that `REFERENCES` the Basetype (the parent)
- The half-circle indicates the Subtype relationship for the entire set
- The Primary Key in every Subtype is identical to the Basetype (as per Single Child, above)
- The Cardinality for the entire set of Subtypes, it is always 1::1, as shown (right)
- The Predicate & Cardinality are not displayed in IDEF1X (they need not be, if the above is understood).



Four scenarios relating to possible Subtypes are differentiated and discussed.

1.2.1 Caveat

- There are some idiots, unfortunately in teaching positions, who suggest that the relationship between the Basetype and the Subtype is not a relationship. Perhaps it is cream cheese. One only needs to check the SQL required to establish the `FOREIGN KEY`, in order to determine the idiocy of their claims.
- The Basetype is not a "supertype", as some writers suggest. The *super* means something technical and specific: the flattened view (derived Relation) of the Basetype and *all* the Subtypes, which is quite different to the form of the Basetype. These are the same idiots who implement Record Filing Systems, labelled as "relational", in violent breach of the *Relational Model*, all while professing knowledge of the *RM*. They are clueless re their loss.
- "Superkeys" or "distributed keys" are the imbecile's method of a *partial* implementation of Subtypes. They double the number of indices, and do not provide the ordinary Referential Integrity or Relational power that the correct method provides. They are simply not required.

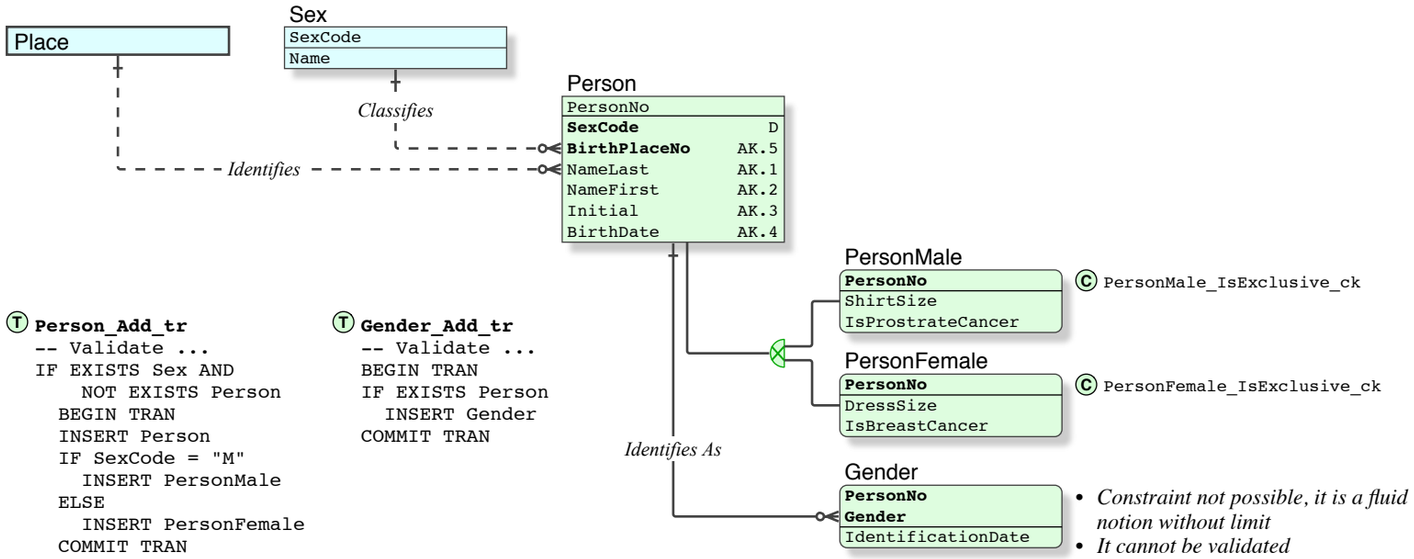
1.3 Transactional Issues

As per the requirements of the **Open Architecture Standard**:

- All constraints on the data must be declared and enforced within the database only. The database must be independent of any application. Note that the database is a single recoverable unit, it may be accessed (Read/Write) by more than one application, and it is certainly accessed (Read only) by multiple report tools.
- A Transaction has the purpose of maintaining data **Integrity**, controls multiple-row database **Consistency**, ensures that the update is **Atomic**. That is three of the four facilities in ACID Transactions. **Durability** is provided by the platform (not the database definition or program code).
- From the very first version, SQL provided ACID Transactions and syntax, all SQL Platforms have it (excluding massive program suites such as Orable; PissGress; etc, that use the term "sql" fraudulently).
- Thus Transactions are a form of Constraint, usually implemented as stored procs within the database.
  - The set of Transactions constitute the **Database API**.
- All updates to the database must be via Transactions only, direct `INSERT/UPDATE/DELETE` to the tables are prohibited (simply not `GRANT`ed). Otherwise, the data container is not merely "closed architecture", it is an unsecured free-for-all, a non-database.

2 Exclusive Subtype

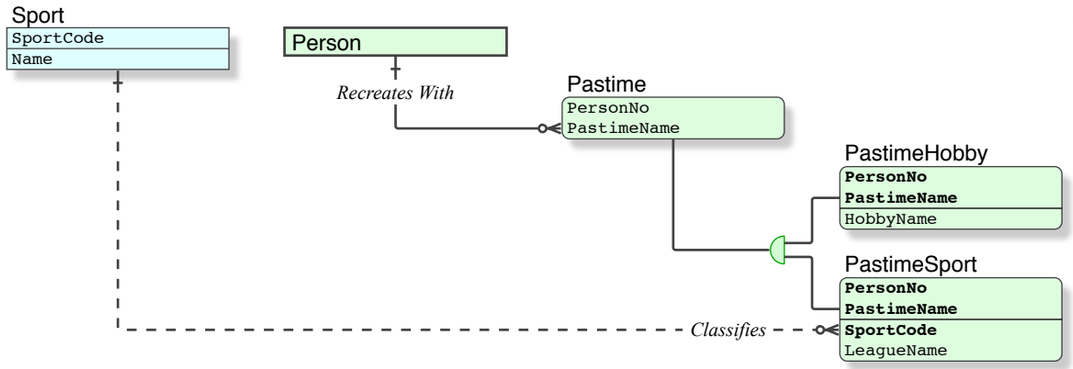
- For each Basetype, there is exactly one Subtype
- A Person is either Male or Female, not both, and not without a Sex
- The Basetype and Subtype are a single Logical unit, they are inserted together, in a single Transaction
- The Basetype must have a **Discriminator** (D) `SexCode`, which is constrained by either a Foreign Key reference or a Rule.
- The exclusivity is constrained via a **`CHECK` Constraint** that executes a **simple Function**.



- *Constraint not possible, it is a fluid notion without limit*
- *It cannot be validated*

3 Non-exclusive Subtype

- For each Basetype, there is at least one Subtype
- A Pastime is one Hobby, or one Sport, or both, not neither
- The Basetype and Subtype are a single Logical unit, they are inserted together, in a single Transaction
- No Discriminator, Subtype existence determination is via a successful `JOIN`.
- The non-exclusivity does not need a constraint.




```
PastimeHobby_Add_tr
-- Validate ...
IF EXISTS Person AND
    NOT EXISTS PastimeHobby
  BEGIN TRAN
  IF NOT EXISTS Pastime
    INSERT Pastime
    INSERT PastimeHobby
  COMMIT TRAN
PastimeSport_Add_tr
-- Validate ...
IF EXISTS Person AND
    NOT EXISTS PastimeSport
  BEGIN TRAN
  IF NOT EXISTS Pastime
    INSERT Pastime
    INSERT PastimeSport
  COMMIT TRAN
```

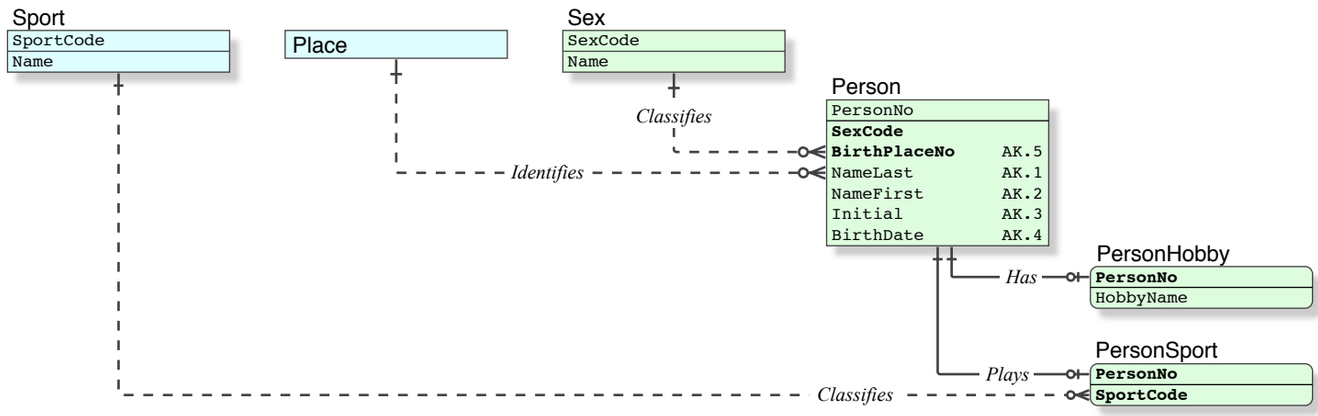
- In the example above the Transaction would be named for the Subtype, eg. `PastimeHobby_Add_tr`; `PastimeSport_Add_tr`. It would insert the Basetype first, and the Subtype second. There would be no `Pastime_Add_tr`.
- Control of *combinations* of Subtypes in a single set should not be required. Simple existence checks in the inserting Transaction should suffice.
  - Note that the need for control of Subtype combinations is an indication of Normalisation errors: if the data was Normalised fully and completely, the dependencies would be more explicit, and the need for such control would disappear
  - For those who insist on a large number of Subtypes that need to be controlled, use a `CHECK` Constraint that calls a simple Function (such as above) that contains the controlling code. An example of the incorrect method will not be provided.

The example given (`Person`) demonstrates the single valid reason for the use of a surrogate: the natural key has become too wide to carry in the child tables. It nevertheless remains a **Relational Breach**.

IDEF1X Notation

4 Optional Attribute[Group], Not Subtype

- If the Basetype can exist without any of the Subtypes, then it is not a Basetype::Subtype Relation, it is an ordinary Relation, which is an optional child (1::0-1)
- Zero or one row for an Optional Attribute (or group of Optional Attributes that are treated together)
- Person has one Sport, or one Hobby, or both, or neither
- The parent and Option are inserted independently, usually the Option is a progression, inserted somewhat after the parent
- The Relation between the parent and the child is, well, an ordinary 1 to 0-or-1 Relation.
- The Primary Key in the child is identical to the Primary Key in the parent, enforcing the max 1 child rule.



5 Mandatory Attribute[Group], Not Subtype

- If the Basetype can exist without any of the Subtypes, then it is not a Basetype::Subtype Relation, it is an ordinary Relation
- If the subject attribute is mandatory, the Cardinality is 1::1. Now in the normal case (Codd's 3NF, not the ever-changing asylum definitions), the subject attribute must be in the parent table. However, there are cases that require advanced Normalisation, be assured that they exist.

  Once the power of Relational Integrity (as distinct from Referential Integrity), which is obtained through the hierarchic Relational Key, is appreciated, an entire level of data integrity and consistency, which was previously not possible, is opened.
- Here two mandatory attributes (`OriginCode`; `DestinationCode`) which would normaly be located in `Flight` are elevated to discrete Facts (Relational Keys) established as `FlightOrigin` and `FlightDestination`, each of which then supplies Relational Integrity to a set of child tables below it.
- One row for a mandatory attribute or group that is treated together.