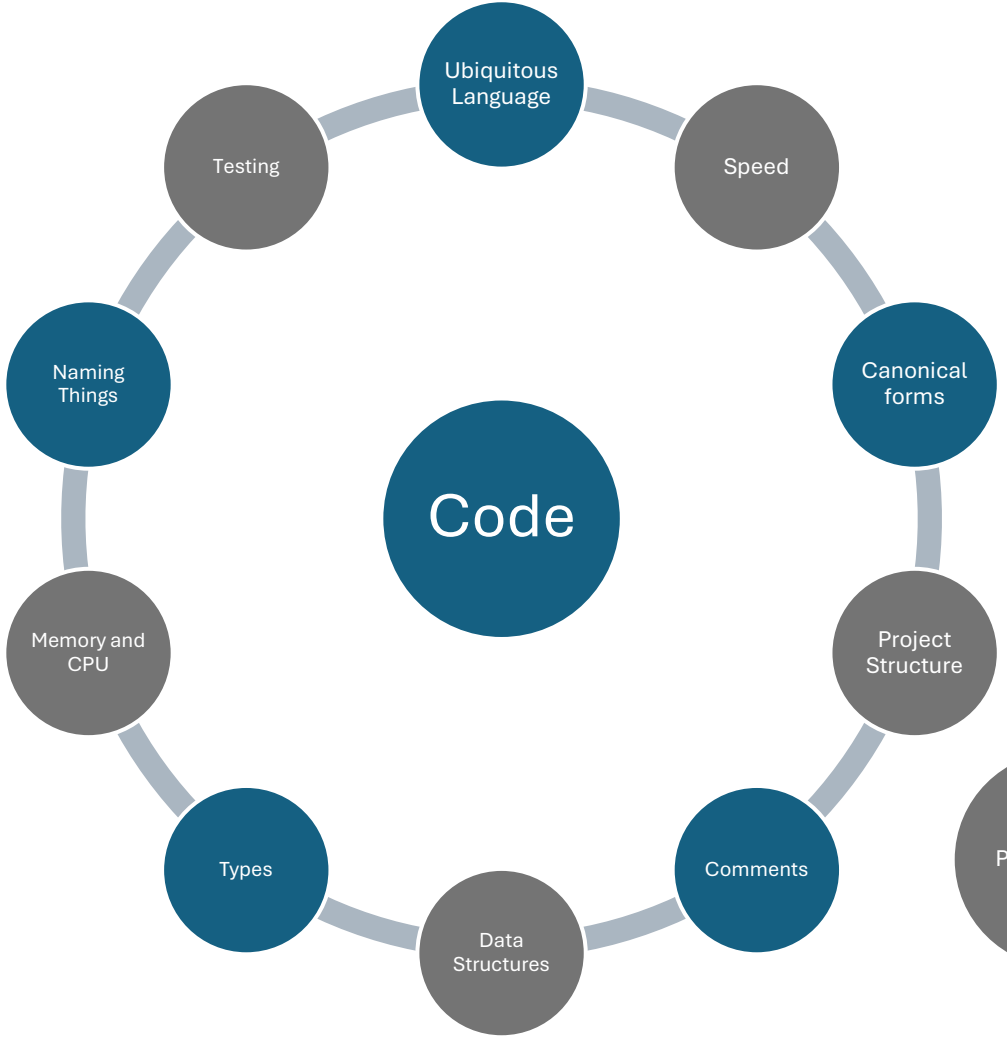
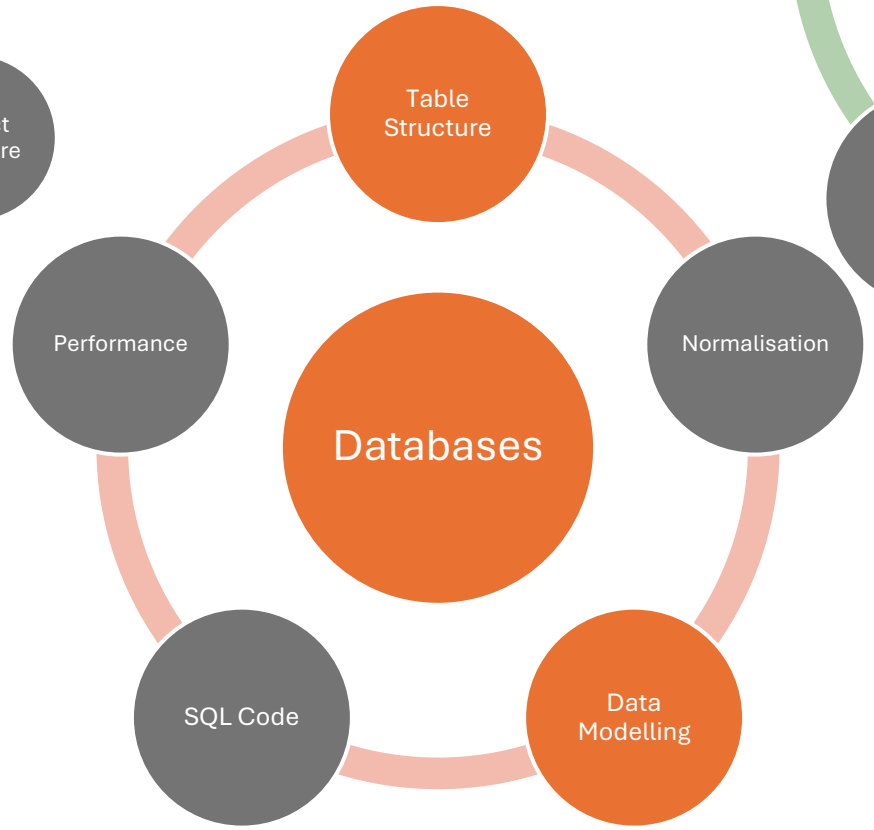
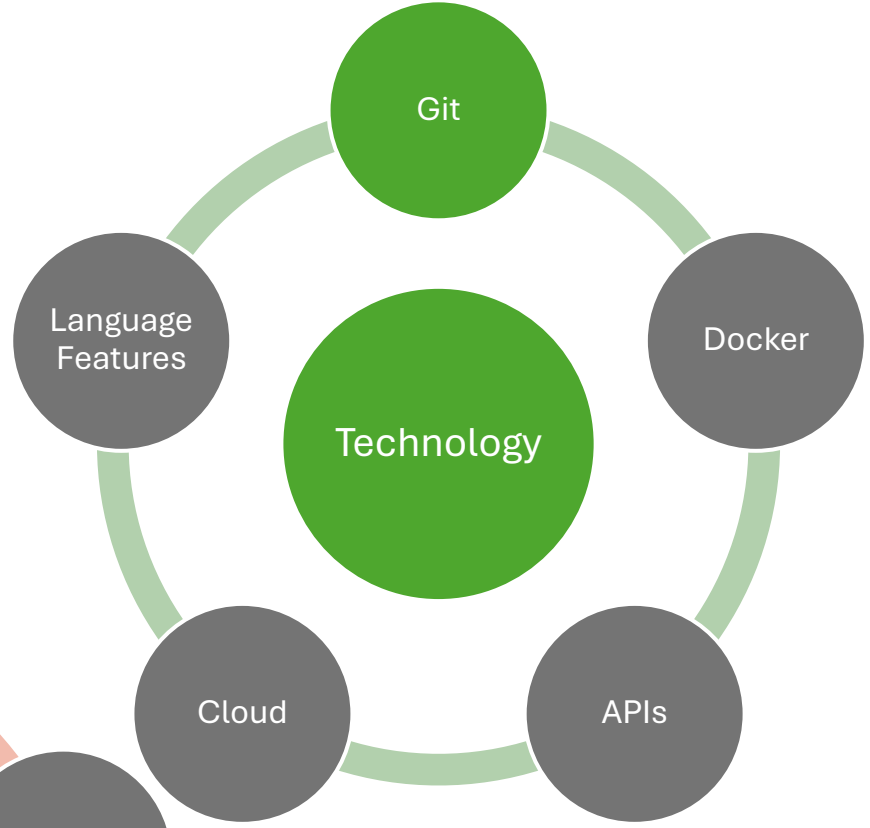
Building Systems That Last

Thomas Hamilton - λ data. t data

CAE September 2024


Code
Ubiquitous Language
Speed
Canonical forms
Project Structure
Comments
Data Structures
Types
Memory and CPU
Naming Things
Testing


Databases
Table Structure
Normalisation
Data Modelling
SQL Code
Performance


Technology
Git
Docker
APIs
Cloud
Language Features

The Ubiquitous Language

Concept in "Domain Driven Design"

- "a set of unambiguous vocabulary shared by all members and stakeholders"

- **Loose language comes with frictional costs!**
  - One concept *one* name *everywhere*
  - Plural vs singular
  - Abbreviations vs full names

Insurance hates it

- Class, class of business, line of business, LOB, MI Class
- Year of Account, Underwriting Year
- Correlations vs dependencies
- Currency vs Currency Code
- Country Code vs Country Code
- Excess vs deductible
- Net vs net net vs net net net
- Biannual business shuffle
- LR vs LR (ratio vs reserve)
- Etc.

The Ubiquitous Language

```
SELECT
    ReservingCurrency,
    ClaimRefs,
    Code,
    Inc,
    TransactionCurrencyCode,
    CMC,
    PP
FROM dbo.Data
WHERE
    SN = 2003
```

```
SELECT
    ReservingCurrency,
    ClaimReference,
    CountryCode,
    IncurredAmount,
    TransactionCurrency,
    CapitalModellingClass,
    PaymentPeriod
FROM dbo.Data
WHERE
    SyndicateNumber = 2003
```

Types

Type	Description	R	Python	VBA	Example
Double	Numbers to 15 significant figures $-1.8 \times 10^{308} \leq x \leq 1.8 \times 10^{308}$	double	float	double	1.2, -1.3e8, 10
Int, Long	Integers made from 32 or 64 bits $-10^{9} \leq x \leq 10^{9}$ or $-10^{18} \leq x \leq 10^{18}$	integer	int	Long or LongLong	0, 100, -314159265
String	Sequence of characters	character	str	String	“This is a string…”
List	Growable sequence of items accessible by their integer index (e.g. ListOfStuff[0], ListOfStuff[10])	c() or list()	list()	Collection or Array	[1, 2, 3.5] [“Gumbel”, “Gaussian”] 2010:2021
Dictionary	Collection of key-value pairs, accessible by their key value (e.g. Dictionary[“Stone”])	c() or list()	dict()	Dictionary	{“Stone”: 0.4, “Brick”: 0.9, “Timber”: 2.4, “Thatch”: 8}

Human Computer

Types

Inflation (rate change) adjust historic premium

Human

$$\$100 \times (1 + 0.03) \times (1 + 0.05) \times \cdots \times (1 + 0.12) = \$156$$

Maths

```
premium: double, inflation: list[double], inflationStartYear: int, asAtYear: int
```

```
premium: list[double], inceptionDates: list[int], expiryDates: list[int],
inflationWeights: dict[string, double], inflationIndices: dict[string, list[double]],
inflationStartYears: dict[string, int], asAtDate: int
```

Computer

Comments


Farringdon Road
Buses towards
Blackfriars and Old Kent Road
Stop


Farringdon Road
Buses towards
Blackfriars and Old Kent Road
Stop
Cowcross Street
Turnmill Street
PRET

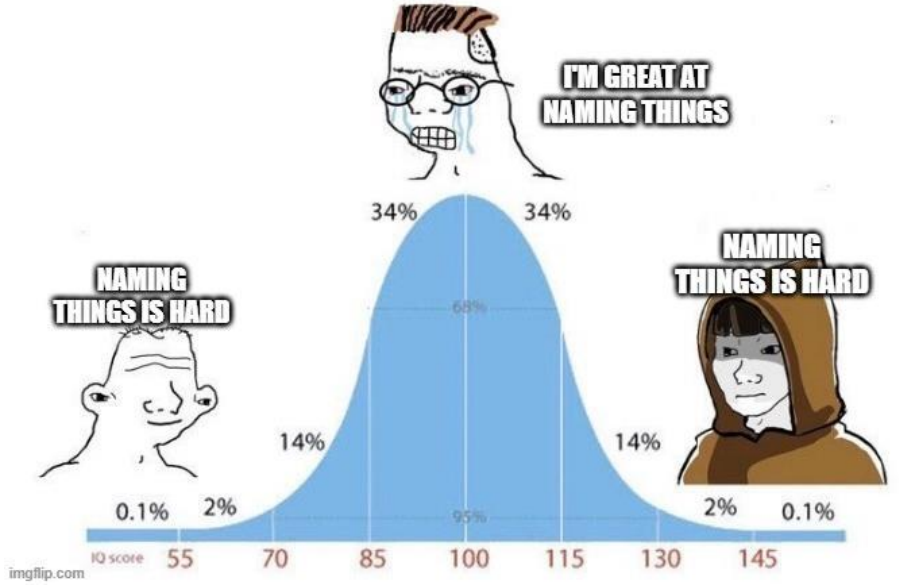

TURNMILL STREET
Jarvis Keller


COWCROSS STREET

Naming things

"...the ratio of time spent reading versus writing is well over 10 to 1...[Therefore,] making it easy to read makes it easier to write."

— Robert C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship



1983 – C Standard Library

`atoi, atoll` – "ASCII to Integer/LongLong etc."

`strcspn` – "max length of initial string segment of first argument not containing characters from second argument"

1995 – Java

`parseInt, parseLong`

`strcspn` – no direct equivalent

Naming things

- Tells you what an object is
  - Count, Min, Max, Sum
  - Claims, Premiums, Factors etc.
  - `claimCounts`, `maxPremiums`
- Describe differences
  - `grossClaims` and `netClaims` instead of `claims1` and `claims2`
- Exclude type information
  - `claimsByCountry` instead of `listClaimsByCountry`
- Avoid abbreviations
  - `PaymentPeriod` instead of `PmtPrd`
- Avoid Acronyms
  - `ParameterisationPeriod` instead of `pp`
- Prefer `Unused` to `Dummy`
  - Include a comment if you're including variables that aren't used – why are they there?

Canonical Forms

…

```
ldf1 = getLdf(p1)
ldf2 = getLdf(p2)
f1 = (pc/p1)^(1/log(p1/p2))
f2 = log(p2) – log(p1)
ldfc = ldf1 * f1 ^ f2
```

…

Canonical Forms

```
…
lossDevelopmentLower = getLossDevelopmentFactor(periodLower)
lossDevelopmentUpper = getLossDevelopmentFactor(periodUpper)
f1 = (periodCurrency / periodLower)^(1 / log(periodLower/periodUpper))
f2 = log(periodUpper) – log(periodLower)
lossDevelopmentCurrent = lossDevelopmentLower * f1 ^ f2
…
```

Canonical Forms

```
…
lossDevLower = getLossDevelopmentFactor(periodLower)
lossDevUpper = getLossDevelopmentFactor(periodUpper)

logDevLower, logDevUpper = log(lossDevlower), log(lossDevUpper)
logPeriodLower, logPeriodUpper = log(periodLower), log(periodUpper)

logPeriodCurrent = log(periodCurrent)

linearInterpolation = (logDevUpper – logDevLower) *
    (logPeriodCurrent – logPeriodLower) / (logPeriodUpper – logPeriodLower)

lossDevelopmentCurrent = exp(logDevLower + linearInterpolation)
…
```

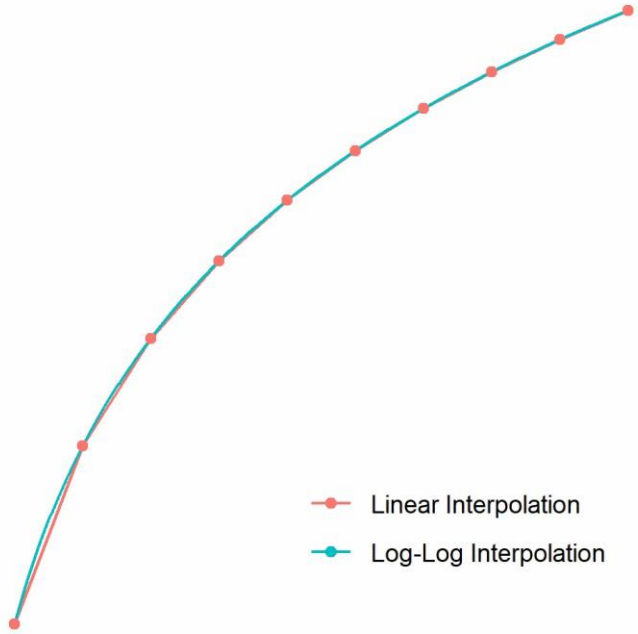
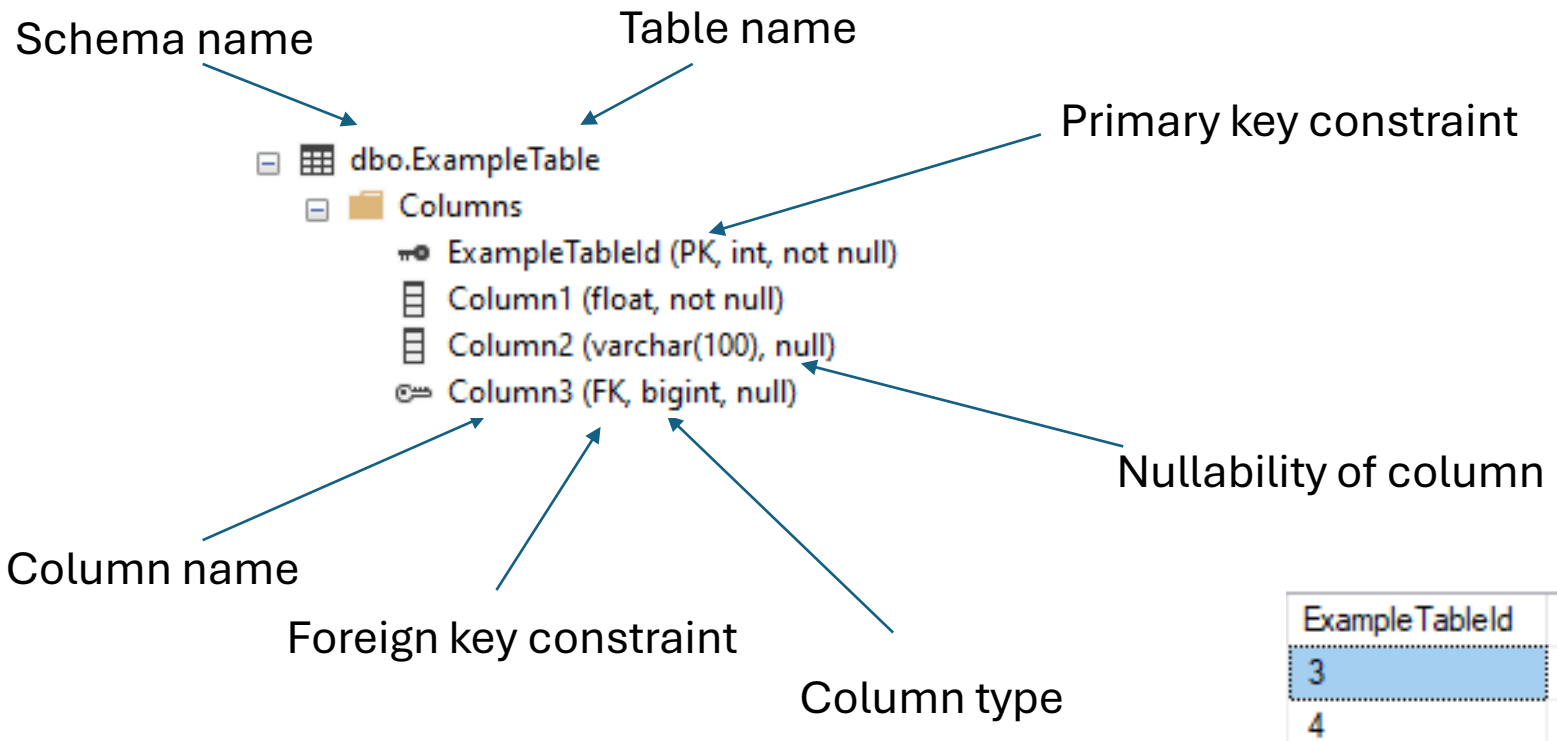
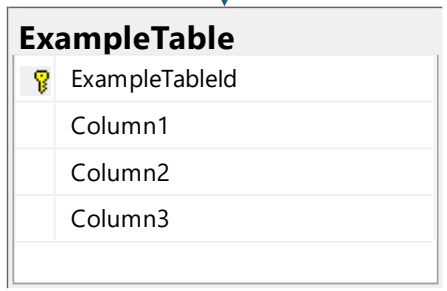


Table Structure



Database Diagram View



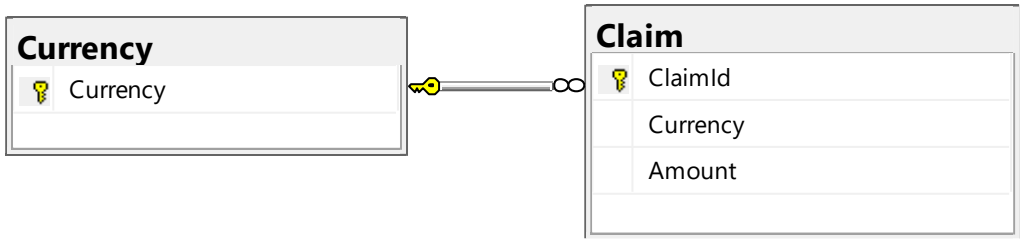
ExampleTableId	Column1	Column2	Column3
3	100.5	Hello GIRO!	NULL
4	-10000	NULL	10000000000000
5	2.71828	I'm a table...	0

Foreign Keys

```
CREATE TABLE dbo.Claim (
    ClaimId  BIGINT PRIMARY KEY IDENTITY,
    Currency VARCHAR(3) REFERENCES dbo.Currency(Currency),
    Amount   FLOAT NOT NULL
)
```

	Currency
1	CAD
2	EUR
3	GBP
4	JPY
5	KRW
6	MXN
7	USD

	ClaimId	Currency	Amount
1	1	USD	100
2	2	USD	200
3	3	GBP	89
4	4	CAD	87
5	5	JPY	1200
6	6	KRW	8299
7	7	MXN	10020



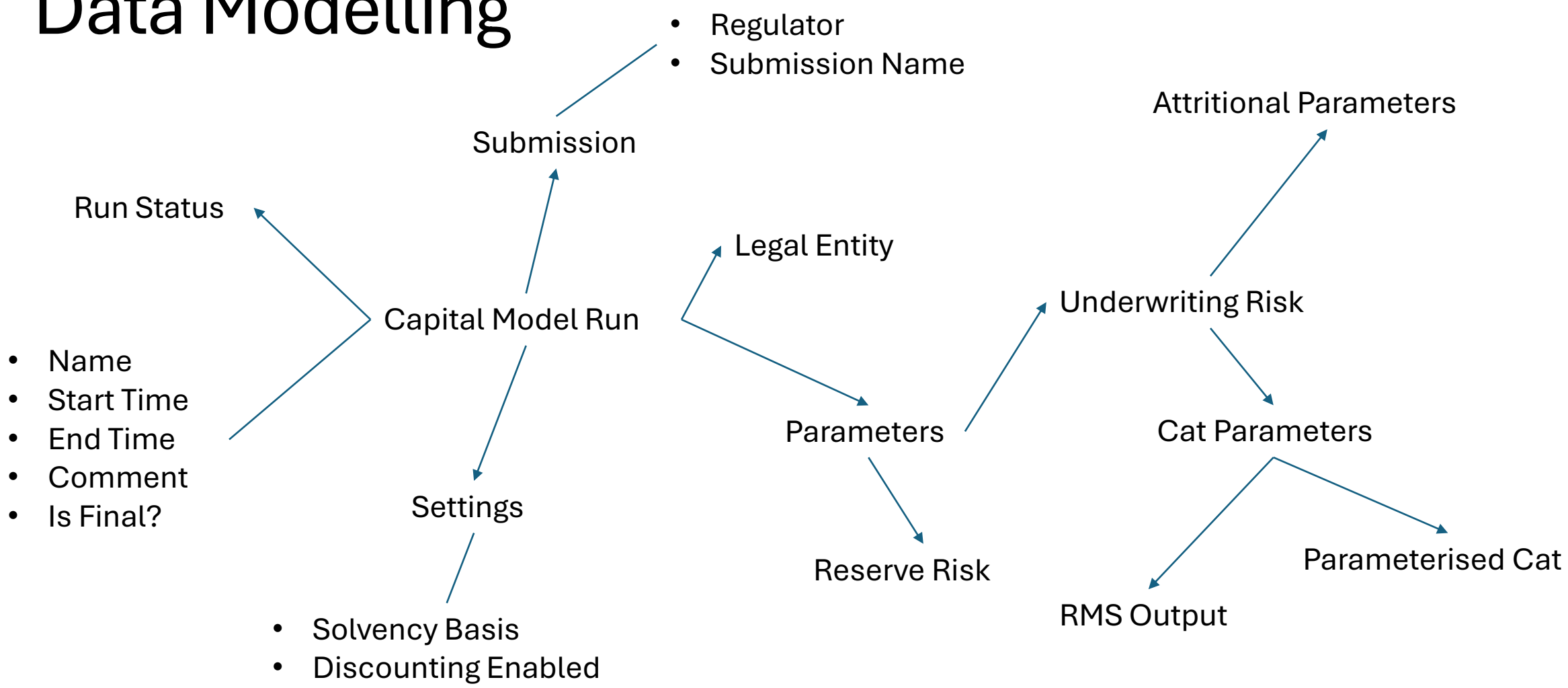

```
36  INSERT INTO dbo.Claim (Currency, Amount) VALUES ('CAR', 100)
```

100 %

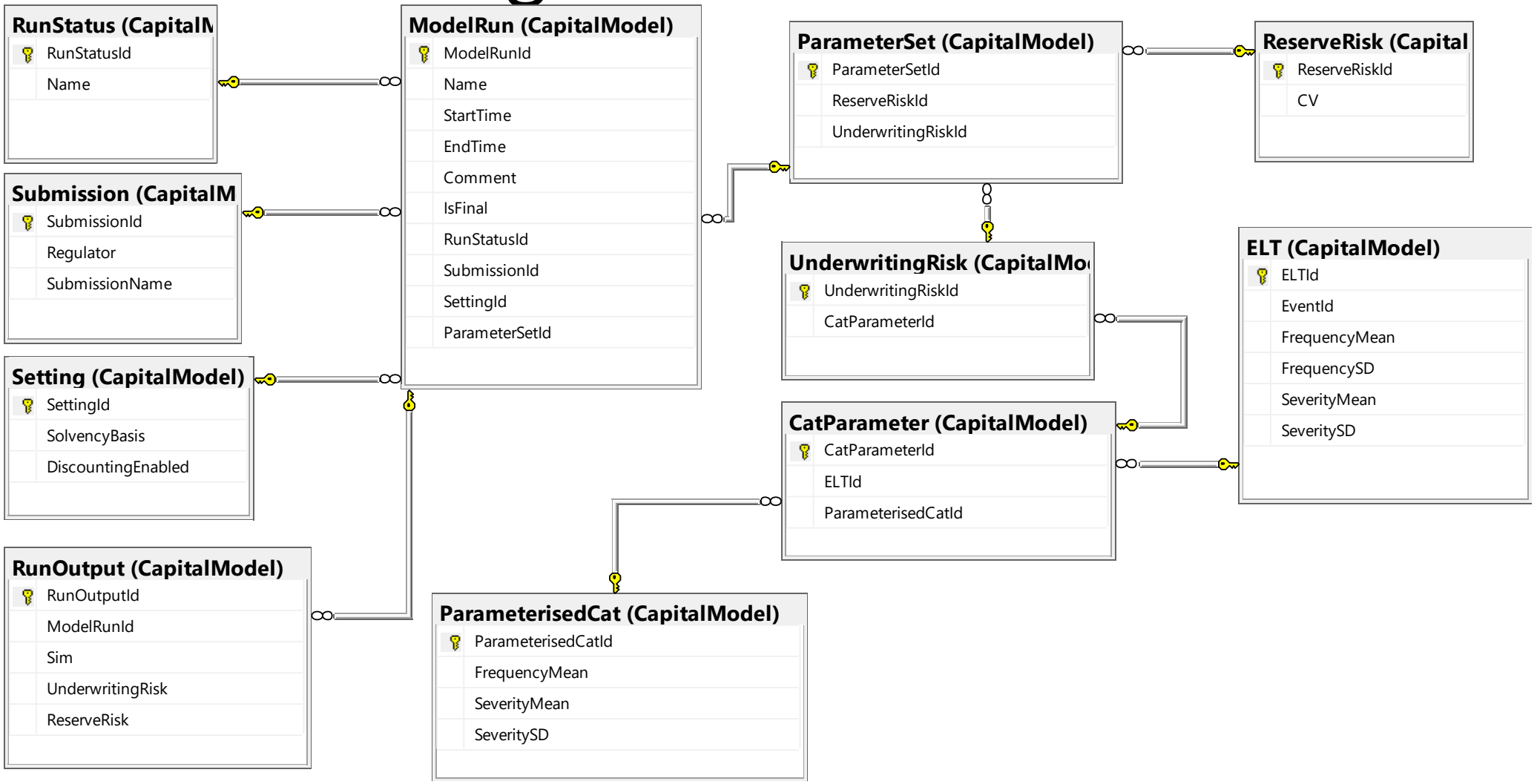
Messages

```
Msg 547, Level 16, State 0, Line 36
The INSERT statement conflicted with the FOREIGN KEY constraint "FK__Claim__Currency__3864608B". The conflict occurred in database "Test", table "dbo.Currency", column 'Currency'.
The statement has been terminated.
```

Data Modelling



Data Modelling



Git

- Project v2.30
- Project v2.29
- Project v2.28
- Project v2.27
- Project v2.26
- Project v2.25_SyndicateSpecial
- Project v2.25_FinalRelease_v2
- Project v2.25_FinalRelease
- Project v2.25
- Project v2.24
- Project v2.23_Patch1_RK
- Project v2.23_Patch1
- Project v2.23
- Project v2.22
- Project v2.21
- Project v2.19
- Project v2.18_FixJohn
- Project v2.18
- Project v2.17
- Project v2.16_ForBoard
- Project v2.16
- Project v2.15

Add cashflowMeanTerm calculation
tomhamiltonlambda committed now

Changing from Binary to Text format
tomhamiltonlambda committed 2 minutes ago

Discounting Fix ...
tomhamiltonlambda committed 13 minutes ago

Add discounting functionality
tomhamiltonlambda committed 14 minutes ago

Add bullet points to README
tomhamiltonlambda committed 16 minutes ago

6 ■■■■■ Modules/Discounting.R

```
@@ -2,3 +2,9 @@
2  discountCashflows <- function(cashflows, discountFactors) {
3      return(cashflows * discountFactors)
4  }
```

```
2  discountCashflows <- function(cashflows, discountFactors) {
3      return(cashflows * discountFactors)
4  }
5  +
6  + cashflowMeanTerm <- function(cashflows) {
7  +     cashflowPeriods <- 1:length(cashflows)
8  +     weightedPeriods <- cashflowPeriods * cashflows
9  +     return(sum(weightedPeriods) / sum(cashflows))
10 + }
```

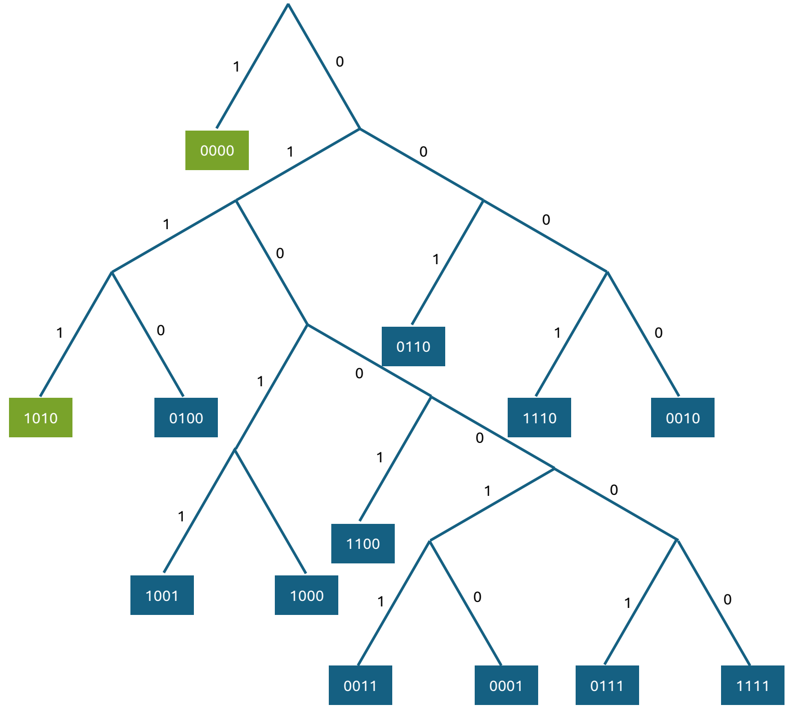
Learn more @ https://learngitbranching.js.org/

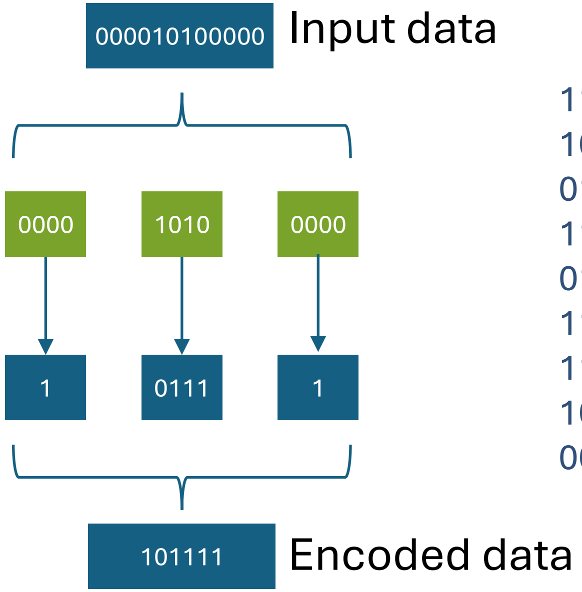
Information Theory

00000000110011100000000010100110000000001001011000000000010011100000000010000110000000001010111000000000001011100000000
01100011000000000100000100000000000000010000000000011001100000000111101100000000000000001000000000100111100000000001011
10000000000001101100000000101011100000000001100011000000000100001100000000011000100000000000000010000000000010011000000
00001110110000000000100001100000000000001000000000010100110000000000101110000000000010101110000000000010111000000001001
01100000000000010111000000001100111000000000011101100000000010010010000000000000001000000000010100110000000000010110000
0000000101010

Length 608

Length 373





110100100011101110011101011001110110000111010
100011101110001110000000111010010011101010000
011101101100100111010000000111101101101011000
111000000011101000110011101110001110100100111
010100011100100001110110110000001110100001001
110101000111101101101110011100000001110111000
111000000011101011001110000000111010010001110
100001001110101100001110110110111001110100010
0011100000111