



Making Sense of Schema-on-Read

Modeling JSON

KENT GRAZIANO, CHIEF TECHNICAL EVANGELIST |  [KentGraziano](#)

About me



- Chief Technical Evangelist, Snowflake Computing
- Oracle ACE Director, Alumni (DW/BI)
- OakTable Network
- Blogger – [The Data Warrior](#)
- Certified Data Vault Master and DV 2.0 Practitioner
- Former Member: Boulder BI Brain Trust (#BBBT)
- Member: DAMA Houston & DAMA International
- Data Architecture and Data Warehouse Specialist
 - 30+ years in IT
 - 25+ years of Oracle-related work
 - 20+ years of data warehousing experience
- Author & Co-Author of a bunch of books (Amazon)
- Past-President of ODTUG and Rocky Mountain Oracle User Group

3 years in stealth + 3+ years GA

Founded 2012 by industry veterans with over 120 database patents



First customers 2014, general availability 2015



Over \$920M in venture funding from leading investors



1200+ employees
Over 2000 customers today

Fun facts:

Queries processed in Snowflake per day:

100 million

Largest single table:

68 trillion rows

Largest number of tables single DB:

200,000

Single customer most data:

> 40PB

Single customer most users:

> 10,000



AGENDA

- Schema-on-Read vs Schema-on-Write
- Why we still need data modeling
- What is JSON?
- Example JSON #1
 - Simple 3NF model
 - Simple Data Vault model
- Example JSON #2
 - 3NF model
 - Data Vault model



DEFINING TERMS

Schema-on-Read

- Popularized in document stores and NoSQL dbs
- No upfront modeling
- No predefined structure
- Called semi-structured or flexible-structure data
 - Can change contents and structure over time
- Load & Go
 - Agile!



DEFINING TERMS

Schema-on-Write

- What we do in RDBMS today
- Requires knowing the structure in advance
- Upfront modeling & table design required
- Must map source data to the database tables
- ETL/ELT may break if the source data changes



163 Zettabytes by 2025



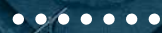
Web



3rd party apps



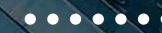
Mobile



Enterprise apps



ERP



IoT

It's not the data itself



it's how you take full advantage of the insight it provides

Who needs data modeling anyway?

- We all do!
- To take advantage of all this data, we have to use it
- Schema-on-Read
 - There is a SCHEMA – which means a model!
- To query the data requires knowing the structure
 - Which means the MODEL of the data or “document”
- Few reporting or BI tools can infer the schema
 - So we have to transform it, somehow
 - Load to tables and columns?
 - Expose with a SQL view?



What is JSON?

- Java
- Script
- Object
- Notation

A minimal, readable format for *structuring* data.

It is used primarily to transmit data between a server and a web application, as an alternative to XML



Why worry about JSON?

- There is LOTS of it out there
- JavaScript is popular
- REST API's for IoT & Mobile
- Application and web logs – Social Media
- Self-describing so very portable
- Open datasets published in JSON
 - Data.gov
 - Datasf.org
 - Data.cityofNewYork.us
- Opportunity for analysis!

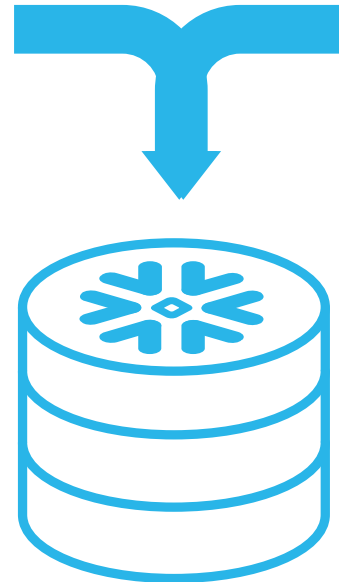


JSON Support with SQL

Structured data

Apple	101.12	250	FIH-2316
Pear	56.22	202	IHO-6912
Orange	98.21	600	WHQ-6090

All Your Data!



Semi-structured data (e.g. JSON, Avro, XML)

```
{  "firstName": "John",  "lastName": "Smith",  "height_cm": 167.64,  "address": {    "streetAddress": "21 2nd Street",    "city": "New York",    "state": "NY",    "postalCode": "10021-3100"  },  "phoneNumbers": [    { "type": "home", "number": "212 555-1234" },    { "type": "office", "number": "646 555-4567" }  ]}
```

select v:lastName::**string** as last_name
from json_demo;

JSON Example #1



```
{
  "colors": [
    {
      "color": "white",
      "category": "hue",
      "type": "primary",
      "code": { "rgba": [255,255,255,1],
                "hex": "#FFFFFF"
              }
    },
    {
      "color": "green",
      "category": "hue",
      "type": "secondary",
      "code": { "rgba": [0,255,0,1],
                "hex": "#0F0"
              }
    }
  ]
}
```

Key : Value

```
"color": "white",
"category": "hue",
"type": "primary",
"code": { "rgba": [255,255,255,1],
          "hex": "#FFFFFF"
        }
},
```

```
"color": "green",
"category": "hue",
"type": "secondary",
"code": { "rgba": [0,255,0,1],
          "hex": "#0F0"
        }
},
```

This is a JSON Document
Enclosed by { }

Elements are Key-Value Pairs

Elements may have nested Keys
Delineated by more { }

Some Values may be Arrays
Delineated by []

JSON as 3NF – Logical Model

```
"colors": [  
  {  
    "color": "white",  
    "category": "hue",  
    "type": "primary",  
    "code": {  
      "rgba": [255,255,255,1],  
      "hex": "#FFFFFF"  
    }  
  }  
]
```

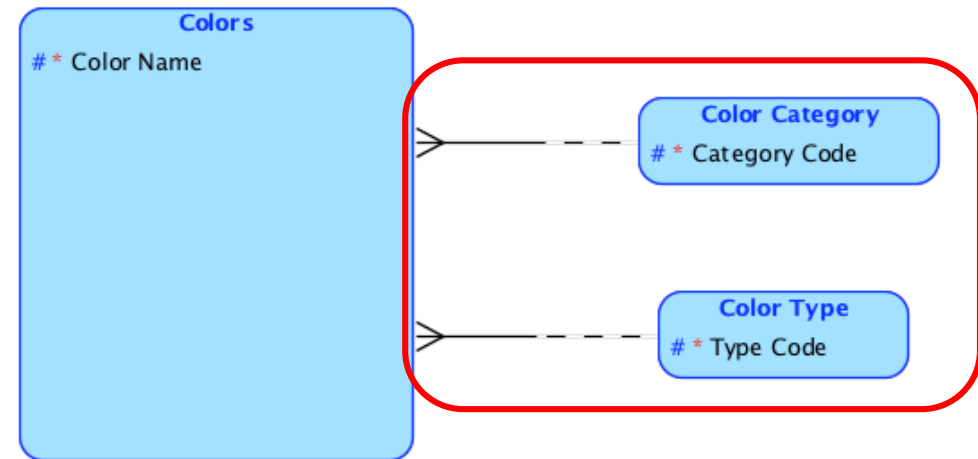
Diagram:	Logical - 3NF
Author:	kgraziano
Created on:	2018-02-04 20:37:59 UTC
Modified on:	2018-02-04 20:37:59 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	Logical



JSON as 3NF – Logical Model

```
"colors": [  
  {  
    "color": "white",  
    "category": "hue",  
    "type": "primary",  
    "code":  
      {  
        "rgba": [255,255,255,1],  
        "hex": "#FFFFFF"  
      }  
  }  
]
```

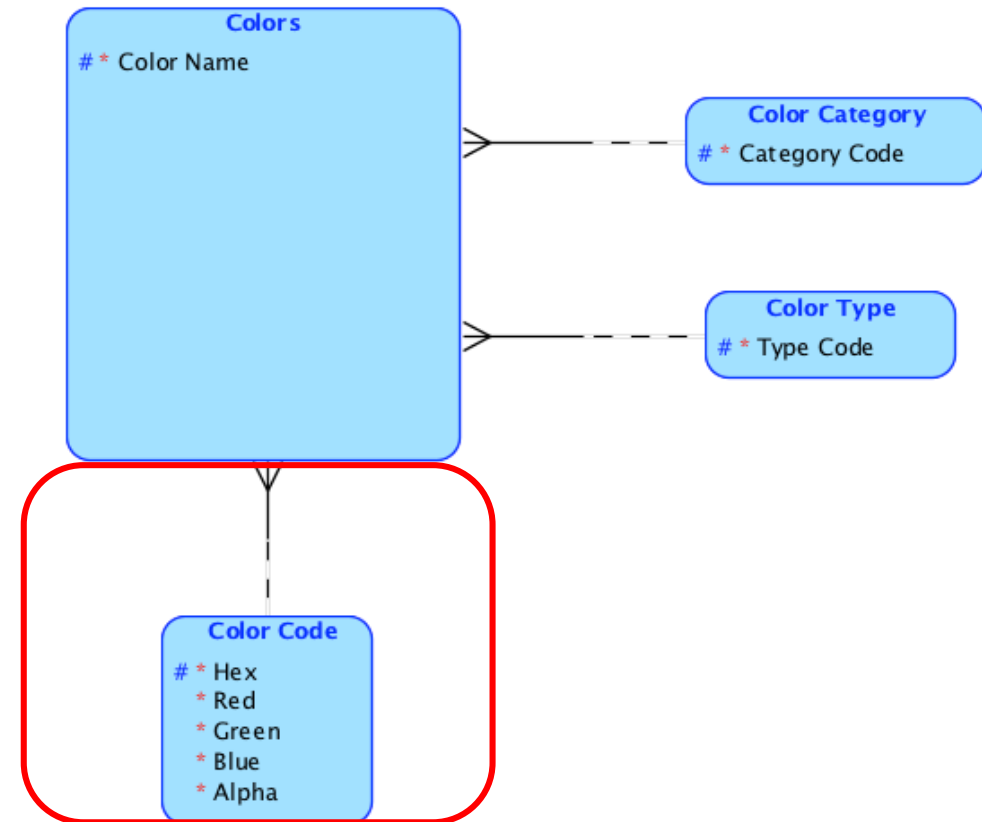
Diagram:	Logical - 3NF
Author:	kgraziano
Created on:	2018-02-04 20:37:59 UTC
Modified on:	2018-02-04 20:37:59 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	Logical



JSON as 3NF – Logical Model

```
"colors": [  
  {  
    "color": "white",  
    "category": "hue",  
    "type": "primary",  
    "code": {  
      "rgba": [255,255,255,1],  
      "hex": "#FFFFFF"  
    }  
  }  
]
```

Diagram:	Logical - 3NF
Author:	kgraziano
Created on:	2018-02-04 20:37:59 UTC
Modified on:	2018-02-04 20:37:59 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	Logical



JSON as 3NF – Schema Model

```
"colors": [  
  {  
    "color": "white",  
    "category": "hue",  
    "type": "primary",  
    "code": {  
      "rgba": [255,255,255,1],  
      "hex": "#FFFFFF"  
    }  
  }  
]
```

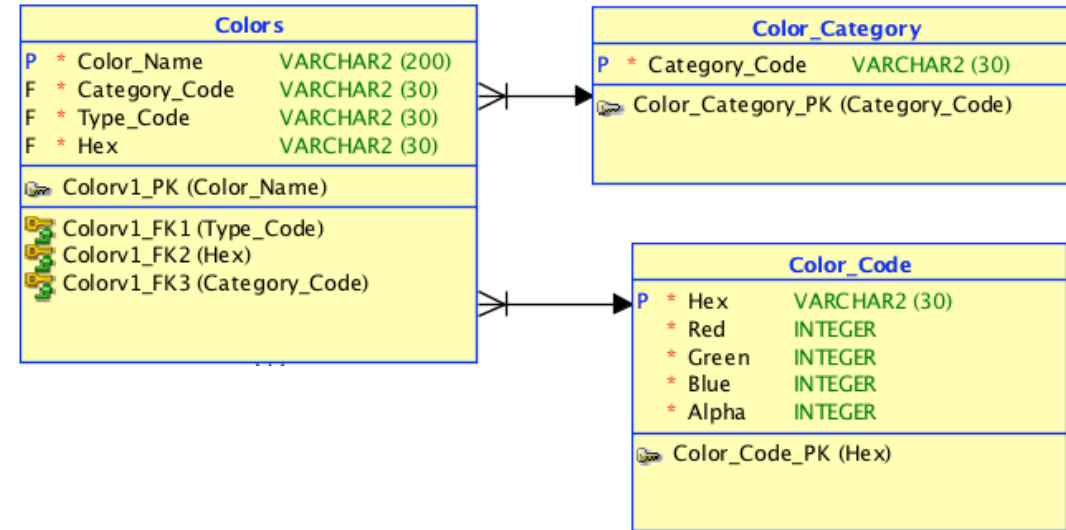
Diagram:	Physical 3NF
Author:	kgraziano
Created on:	2018-02-04 20:48:09 UTC
Modified on:	2018-02-04 20:48:17 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	JSON 3NF

Colors		
P *	Color_Name	VARCHAR2 (200)
F *	Category_Code	VARCHAR2 (30)
F *	Type_Code	VARCHAR2 (30)
F *	Hex	VARCHAR2 (30)
Colorv1_PK (Color_Name)		
Colorv1_FK1 (Type_Code)		
Colorv1_FK2 (Hex)		
Colorv1_FK3 (Category_Code)		

JSON as 3NF – Schema Model

```
"colors": [  
  {  
    "color": "white",  
    "category": "hue",  
    "type": "primary",  
    "code": {  
      "rgba": [255,255,255,1],  
      "hex": "#FFFFFF"  
    }  
  }  
]
```

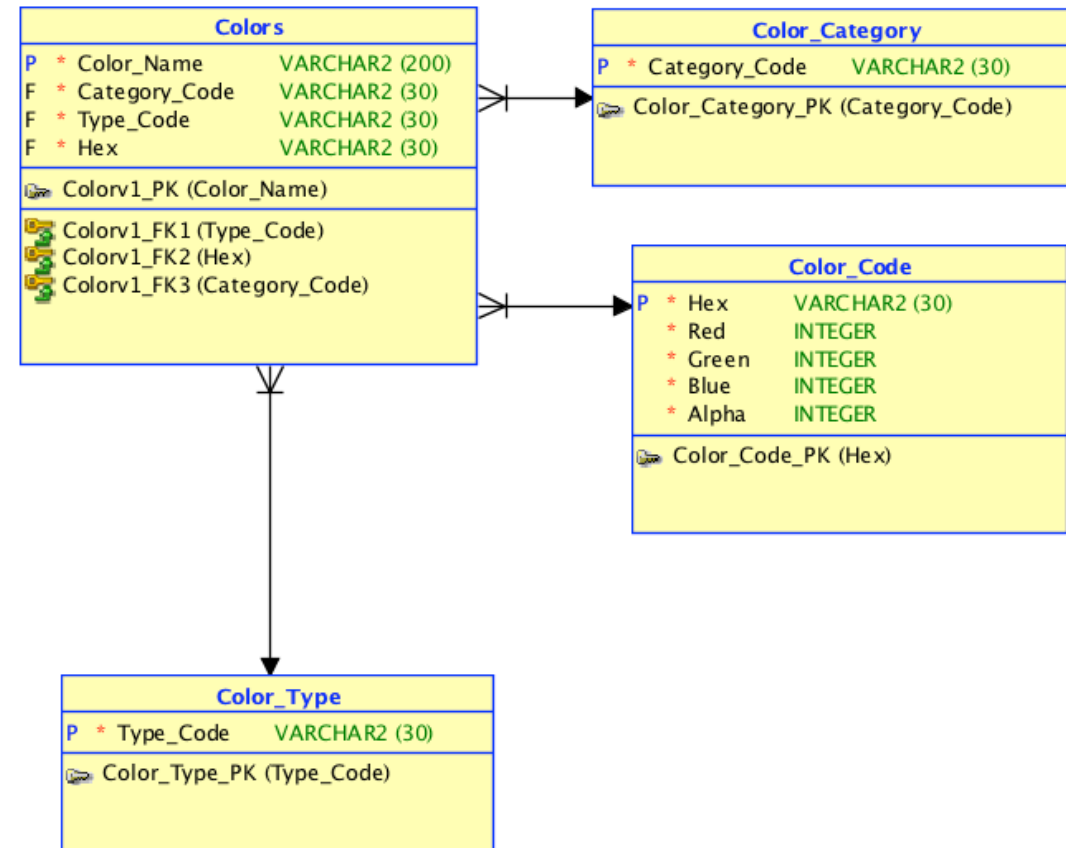
Diagram:	Physical 3NF
Author:	kgraziano
Created on:	2018-02-04 20:48:09 UTC
Modified on:	2018-02-04 20:48:17 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	JSON 3NF



JSON as 3NF – Schema Model

```
"colors": [  
  {  
    "color": "white",  
    "category": "hue",  
    "type": "primary",  
    "code": {  
      "rgba": [255,255,255,1],  
      "hex": "#FFFFFF"  
    }  
  }  
]
```



Diagram:	Physical 3NF
Author:	kgraziano
Created on:	2018-02-04 20:48:09 UTC
Modified on:	2018-02-04 20:48:17 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	JSON 3NF



JSON as Denormalized – Relational Model

```
"colors": [  
  {  
    "color": "white",  
    "category": "hue",  
    "type": "primary",  
    "code":  
      {  
        "rgba": [255,255,255,1],  
        "hex": "#FFFFFF"  
      }  
  }  
]
```

Diagram:	Denormalized
Author:	kgraziano
Created on:	2018-02-04 20:50:48 UTC
Modified on:	2018-02-04 20:50:55 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	JSON 3NF

Color		
P *	Color_Name	VARCHAR2 (200)
*	Category	VARCHAR2 (200)
*	Type	VARCHAR2 (30)
*	RGBA_Code	VARCHAR2 (200)
*	Hex_Code	VARCHAR2 (30)
	Color_PK (Color_Name)	
	Color_UK 1 (Color_Name)	

Data Vault Style

JSON as Data Vault

```
"colors": [  
  {  
    "color": "white",  
    "category": "hue",  
    "type": "primary",  
    "code":  
    {  
      "rgba": [255,255,255,1],  
      "hex": "#FFFFFF"  
    }  
  }  
]
```

Diagram:	Data Vault
Author:	kgraziano
Created on:	2018-02-04 21:33:14 UTC
Modified on:	2018-02-04 21:33:19 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	JSON Data Vault

Hub_Color_Category	
P	Hub_Color_Category_MD5_Key VARCHAR2 (32)
U	Category_Code VARCHAR2 (30)
	LOAD_DTS DATE
	REC_SRC VARCHAR2 (100)
Hub_Color_Category_PK (Hub_Color_Category_MD5_Key)	
Hub_Color_Category_UK1 (Category_Code)	

JSON as Data Vault

```
"colors": [  
  {  
    "color": "white",  
    "category": "hue",  
    "type": "primary",  
    "code": {  
      "rgba": [255,255,255,1],  
      "hex": "#FFFFFF"  
    }  
  }  
]
```

Diagram:	Data Vault
Author:	kgraziano
Created on:	2018-02-04 21:33:14 UTC
Modified on:	2018-02-04 21:33:19 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	JSON Data Vault

H	Hub_Color_Category
P	* Hub_Color_Category_MD5_Key VARCHAR2 (32)
U	* Category_Code VARCHAR2 (30)
	* LOAD_DTS DATE
	* REC_SRC VARCHAR2 (100)
	Hub_Color_Category_PK (Hub_Color_Category_MD5_Key)
	Hub_Color_Category_UK1 (Category_Code)

H	Hub_Color_Type
P	* Hub_Type_MD5_Key VARCHAR2 (32)
U	* Type_Code VARCHAR2 (30)
	* LOAD_DTS DATE
	* REC_SRC VARCHAR2 (100)
	Hub_Color_Type_PK (Hub_Type_MD5_Key)
	Hub_Color_Type_UK1 (Type_Code)

JSON as Data Vault

```
"colors": [  
  {  
    "color": "white",  
    "category": "hue",  
    "type": "primary",  
    "code": {  
      "rgba": [255,255,255,1],  
      "hex": "#FFFFFF"  
    }  
  }  
]
```

Diagram:	Data Vault
Author:	kgraziano
Created on:	2018-02-04 21:33:14 UTC
Modified on:	2018-02-04 21:33:19 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	JSON Data Vault

H Hub_Color_Category	
P *	Hub_Color_Category_MDS_Key VARCHAR2 (32)
U *	Category_Code VARCHAR2 (30)
*	LOAD_DTS DATE
*	REC_SRC VARCHAR2 (100)
Hub_Color_Category_PK (Hub_Color_Category_MDS_Key)	
Hub_Color_Category_UK1 (Category_Code)	

H Hub_Color_Type	
P *	Hub_Type_MDS_Key VARCHAR2 (32)
U *	Type_Code VARCHAR2 (30)
*	LOAD_DTS DATE
*	REC_SRC VARCHAR2 (100)
Hub_Color_Type_PK (Hub_Type_MDS_Key)	
Hub_Color_Type_UK1 (Type_Code)	

H Hub_Color_Code	
P *	Hub_Color_Code_MDS_Key VARCHAR2 (32)
U *	Hex VARCHAR2 (30)
*	LOAD_DTS DATE
*	REC_SRC VARCHAR2 (100)
Hub_Color_Code_PK (Hub_Color_Code_MDS_Key)	
Hub_Color_Code_UK1 (Hex)	

JSON as Data Vault

```
"colors": [  
  {  
    "color": "white",  
    "category": "hue",  
    "type": "primary",  
    "code": {  
      "rgba": [255,255,255,1],  
      "hex": "#FFFFFF"  
    }  
  }  
]
```

Diagram:	Data Vault
Author:	kgraziano
Created on:	2018-02-04 21:33:14 UTC
Modified on:	2018-02-04 21:33:19 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	JSON Data Vault

H	Hub_Color_Category
P	* Hub_Color_Category_MDS_Key VARCHAR2 (32)
U	* Category_Code VARCHAR2 (30)
	* LOAD_DTS DATE
	* REC_SRC VARCHAR2 (100)
	Hub_Color_Category_PK (Hub_Color_Category_MDS_Key)
	Hub_Color_Category_UK1 (Category_Code)

H	Hub_Color_Type
P	* Hub_Type_MDS_Key VARCHAR2 (32)
U	* Type_Code VARCHAR2 (30)
	* LOAD_DTS DATE
	* REC_SRC VARCHAR2 (100)
	Hub_Color_Type_PK (Hub_Type_MDS_Key)
	Hub_Color_Type_UK1 (Type_Code)

H	Hub_Color_Code
P	* Hub_Color_Code_MDS_Key VARCHAR2 (32)
U	* Hex VARCHAR2 (30)
	* LOAD_DTS DATE
	* REC_SRC VARCHAR2 (100)
	Hub_Color_Code_PK (Hub_Color_Code_MDS_Key)
	Hub_Color_Code_UK1 (Hex)

S	Sat_Color_Code
PF	* Hub_Color_Code_MDS_Key VARCHAR2 (32)
P	* LOAD_DTS DATE
	* Red INTEGER
	* Green INTEGER
	* Blue INTEGER
	* Alpha INTEGER
	* HASH_DIFF VARCHAR2 (32)
	* REC_SRC VARCHAR2 (100)
	Sat_Color_Code_PK (Hub_Color_Code_MDS_Key, LOAD_DTS)
	Sat_Color_Code_FK0 (Hub_Color_Code_MDS_Key)

JSON as Data Vault

```
"colors": [
```

```
{
  "color": "white",
  "category": "hue",
  "type": "primary",
  "code": {
    "rgba": [255,255,255,1],
    "hex": "#FFFFFF"
  }
}]
```

Diagram:	Data Vault
Author:	kgraziano
Created on:	2018-02-04 21:33:14 UTC
Modified on:	2018-02-04 21:33:19 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	JSON Data Vault

H Hub_Color_Code	
P *	Hub_Color_Code_MDS_Key VARCHAR2 (32)
U *	Hex VARCHAR2 (30)
*	LOAD_DTS DATE
*	REC_SRC VARCHAR2 (100)
Hub_Color_Code_PK (Hub_Color_Code_MDS_Key)	
Hub_Color_Code_UK1 (Hex)	

S Sat_Color_Code	
PF *	Hub_Color_Code_MDS_Key VARCHAR2 (32)
P *	LOAD_DTS DATE
*	Red INTEGER
*	Green INTEGER
*	Blue INTEGER
*	Alpha INTEGER
*	HASH_DIFF VARCHAR2 (32)
*	REC_SRC VARCHAR2 (100)
Sat_Color_Code_PK (Hub_Color_Code_MDS_Key, LOAD_DTS)	
Sat_Color_Code_FK0 (Hub_Color_Code_MDS_Key)	

H Hub_Color_Category	
P *	Hub_Color_Category_MDS_Key VARCHAR2 (32)
U *	Category_Code VARCHAR2 (30)
*	LOAD_DTS DATE
*	REC_SRC VARCHAR2 (100)
Hub_Color_Category_PK (Hub_Color_Category_MDS_Key)	
Hub_Color_Category_UK1 (Category_Code)	

H Hub_Color_Type	
P *	Hub_Type_MDS_Key VARCHAR2 (32)
U *	Type_Code VARCHAR2 (30)
*	LOAD_DTS DATE
*	REC_SRC VARCHAR2 (100)
Hub_Color_Type_PK (Hub_Type_MDS_Key)	
Hub_Color_Type_UK1 (Type_Code)	

L Link_Colors	
P *	Link_MDS_KEY VARCHAR2 (32)
UF *	Hub_Color_Category_MDS_Key VARCHAR2 (32)
UF *	Hub_Type_MDS_Key VARCHAR2 (32)
UF *	Hub_Color_Code_MDS_Key VARCHAR2 (32)
U *	Category_Code VARCHAR2 (30)
U *	Type_Code VARCHAR2 (30)
U *	Hex VARCHAR2 (30)
*	LOAD_DTS VARCHAR2 (30)
*	REC_SRC VARCHAR2 (30)
Link_Colors_PK (Link_MDS_KEY)	
Link_Colors_UK1 (Category_Code, Type_Code, Hex)	
Link_Colors_UK2 (Hub_Color_Category_MDS_Key, Hub_Type_MDS_Key, Hub_Color_Code_MDS_Key)	
Link_Colors_FK6 (Hub_Type_MDS_Key)	
Link_Colors_FK7 (Hub_Color_Category_MDS_Key)	
Link_Colors_FK9 (Hub_Color_Code_MDS_Key)	

JSON as Data Vault

```

"colors": [
{
"color": "white",
"category": "hue",
"type": "primary",
"code":
{
"rgba": [255,255,255,1],
"hex": "#FFFFFF"
}
}
]
    
```

Diagram:	Data Vault
Author:	kgraziano
Created on:	2018-02-04 21:33:14 UTC
Modified on:	2018-02-04 21:33:19 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	JSON Data Vault

H Hub_Color_Code	
P *	Hub_Color_Code_MDS_Key VARCHAR2 (32)
U *	Hex VARCHAR2 (30)
*	LOAD_DTS DATE
*	REC_SRC VARCHAR2 (100)
↳ Hub_Color_Code_PK (Hub_Color_Code_MDS_Key)	
◊	Hub_Color_Code_UK1 (Hex)

S Sat_Color_Code	
PF *	Hub_Color_Code_MDS_Key VARCHAR2 (32)
P *	LOAD_DTS DATE
*	Red INTEGER
*	Green INTEGER
*	Blue INTEGER
*	Alpha INTEGER
*	HASH_DIFF VARCHAR2 (32)
*	REC_SRC VARCHAR2 (100)
↳ Sat_Color_Code_PK (Hub_Color_Code_MDS_Key, LOAD_DTS)	
↳	Sat_Color_Code_FK0 (Hub_Color_Code_MDS_Key)

H Hub_Color_Category	
P *	Hub_Color_Category_MDS_Key VARCHAR2 (32)
U *	Category_Code VARCHAR2 (30)
*	LOAD_DTS DATE
*	REC_SRC VARCHAR2 (100)
↳ Hub_Color_Category_PK (Hub_Color_Category_MDS_Key)	
◊	Hub_Color_Category_UK1 (Category_Code)

H Hub_Color_Type	
P *	Hub_Type_MDS_Key VARCHAR2 (32)
U *	Type_Code VARCHAR2 (30)
*	LOAD_DTS DATE
*	REC_SRC VARCHAR2 (100)
↳ Hub_Color_Type_PK (Hub_Type_MDS_Key)	
◊	Hub_Color_Type_UK1 (Type_Code)

L Link_Colors	
P *	Link_MDS_Key VARCHAR2 (32)
UF *	Hub_Color_Category_MDS_Key VARCHAR2 (32)
UF *	Hub_Type_MDS_Key VARCHAR2 (32)
UF *	Hub_Color_Code_MDS_Key VARCHAR2 (32)
U *	Category_Code VARCHAR2 (30)
U *	Type_Code VARCHAR2 (30)
U *	Hex VARCHAR2 (30)
*	LOAD_DTS VARCHAR2 (30)
*	REC_SRC VARCHAR2 (30)
↳ Link_Colors_PK (Link_MDS_Key)	
◊	Link_Colors_UK1 (Category_Code, Type_Code, Hex)
◊	Link_Colors_UK2 (Hub_Color_Category_MDS_Key, Hub_Type_MDS_Key, Hub_Color_Code_MDS_Key)
↳	Link_Colors_FK6 (Hub_Type_MDS_Key)
↳	Link_Colors_FK7 (Hub_Color_Category_MDS_Key)
↳	Link_Colors_FK9 (Hub_Color_Code_MDS_Key)

S Sat_Colors	
PF *	Link_MDS_Key VARCHAR2 (32)
P *	LOAD_DTS DATE
*	Color_Name VARCHAR2 (200)
*	HASH_DIFF VARCHAR2 (32)
*	REC_SRC VARCHAR2 (100)
↳ Sat_Colors_PK (Link_MDS_Key, LOAD_DTS)	
↳	Sat_Colors_FK0 (Link_MDS_Key)

What if the JSON changes?

- That is the point of schema-on-read
 - No changes to ingest the data
 - NoSQL, Snowflake, Oracle
- Example
 - More attributes on Color Category or Color Type
 - Like “Description”
 - In a 3NF model
 - Add new columns to entities/tables
 - ALTER TABLE required
 - In a Data Vault model
 - Add new Sat tables on existing Hubs
 - CREATE TABLE required
 - No change required to existing tables

JSON Example #2

```
{
  "fullName": "Johnny Appleseed",
  "age": 42,
  "gender": "Male",
  "phoneNumber":
    {
      "areaCode": "415",
      "subscriberNumber": "5551234"
    },
  "children":
    [
      {
        "name": "Jayden",
        "gender": "Male",
        "age": "10" },
      {
        "name": "Emma",
        "gender": "Female",
        "age": "8" },
      {
        "name": "Madelyn",
        "gender": "Female",
        "age": "6" }
    ],

```

Nested Elements

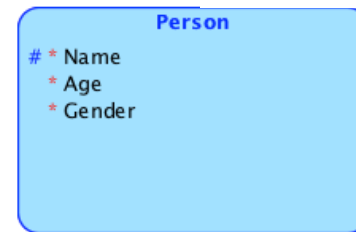
Nested Array of Values,
Within a Nested Array
Of Elements

Nested Array of Elements

```
"citiesLived": [
  {
    "cityName": "London",
    "yearsLived": [ "1989", "1993", "1998", "2002" ]
  },
  {
    "cityName": "San Francisco",
    "yearsLived": [ "1990", "1993", "1998", "2008" ]
  },
  {
    "cityName": "Portland",
    "yearsLived": [ "1993", "1998", "2003", "2005" ]
  }
]
```

JSON as 3NF – Logical Model

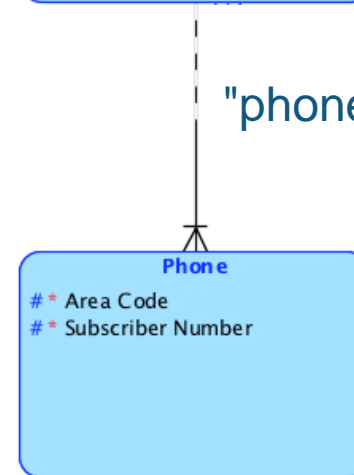
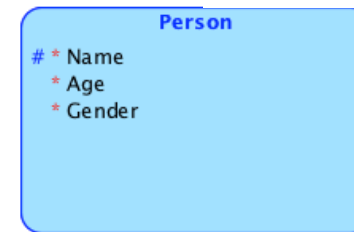
Diagram:	Logical 2 – 3NF
Author:	kgraziano
Created on:	2018-02-05 04:29:13 UTC
Modified on:	2018-02-05 04:29:13 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	Logical



"fullName": "Johnny Appleseed",
"age": 42,
"gender": "Male",

JSON as 3NF – Logical Model

Diagram:	Logical 2 - 3NF
Author:	kgraziano
Created on:	2018-02-05 04:29:13 UTC
Modified on:	2018-02-05 04:29:13 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	Logical

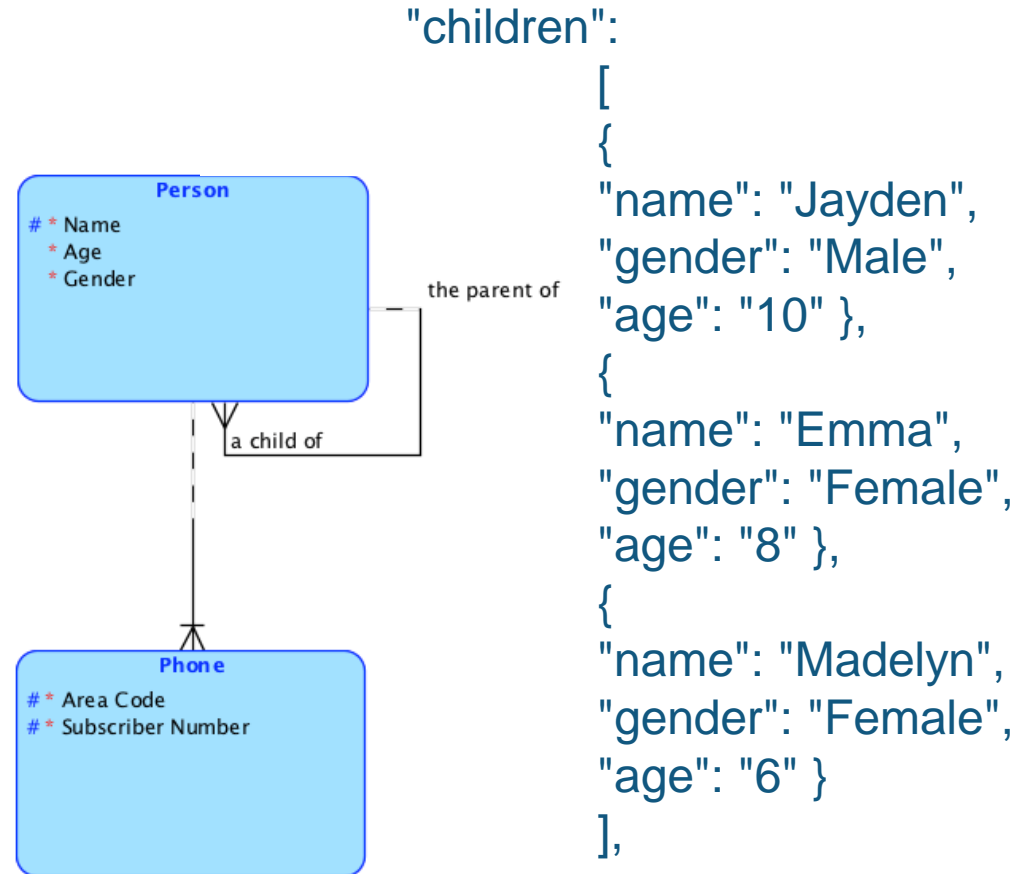


"phoneNumber":

```
{  
  "areaCode": "415",  
  "subscriberNumber": "5551234"  
},
```

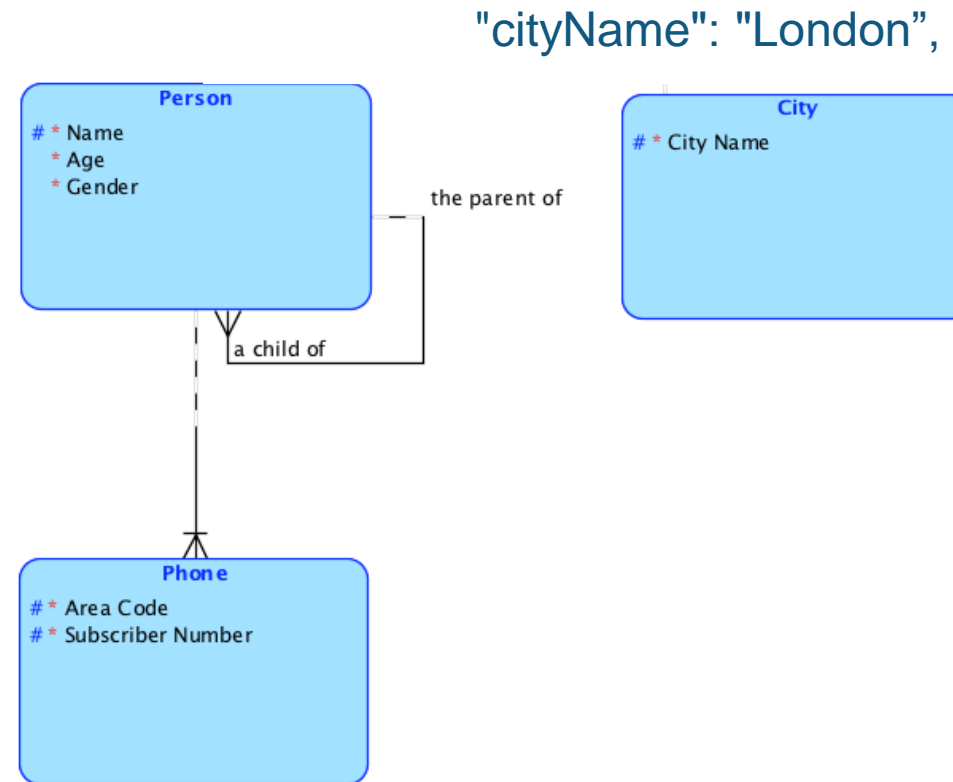
JSON as 3NF – Logical Model

Diagram:	Logical 2 - 3NF
Author:	kgraziano
Created on:	2018-02-05 04:29:13 UTC
Modified on:	2018-02-05 04:29:13 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	Logical



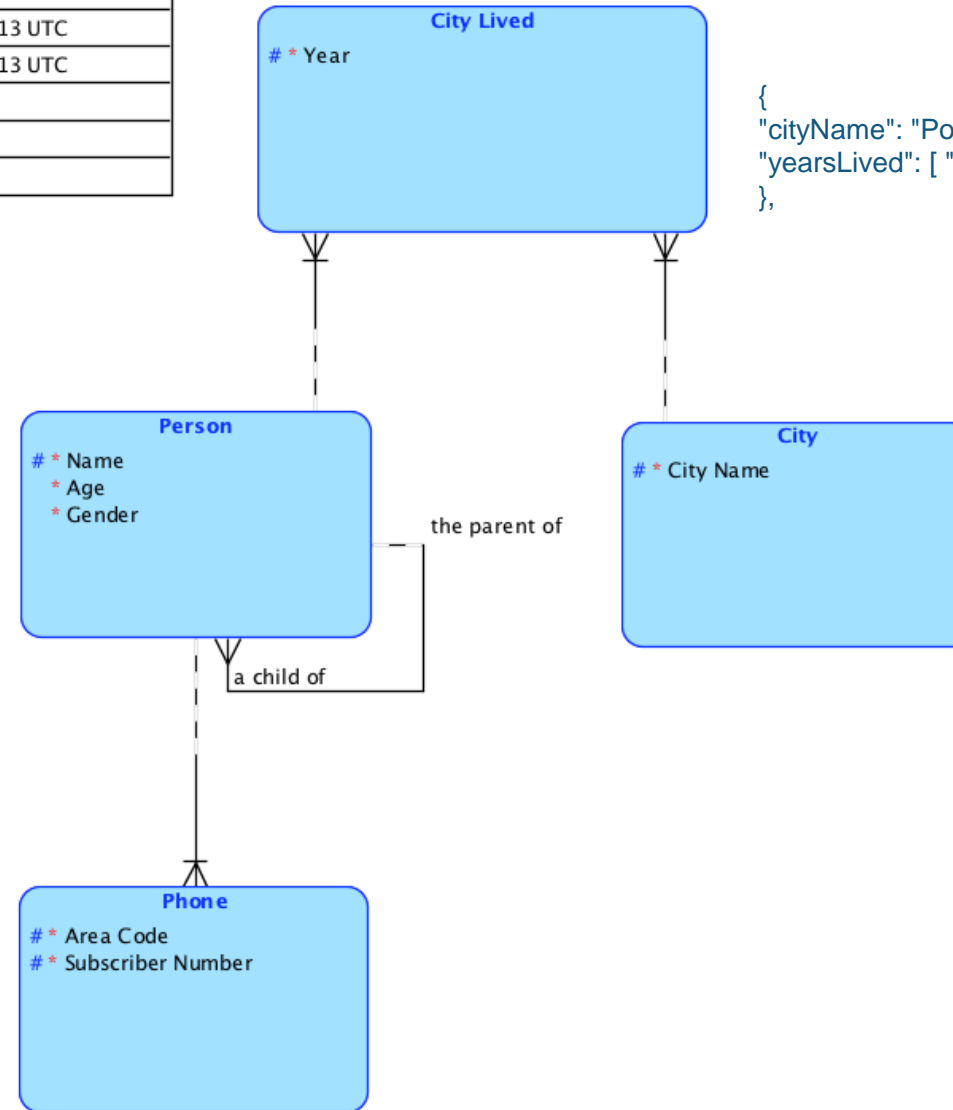
JSON as 3NF – Logical Model

Diagram:	Logical 2 - 3NF
Author:	kgraziano
Created on:	2018-02-05 04:29:13 UTC
Modified on:	2018-02-05 04:29:13 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	Logical



JSON as 3NF – Logical Model

Diagram:	Logical 2 - 3NF
Author:	kgraziano
Created on:	2018-02-05 04:29:13 UTC
Modified on:	2018-02-05 04:29:13 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	Logical

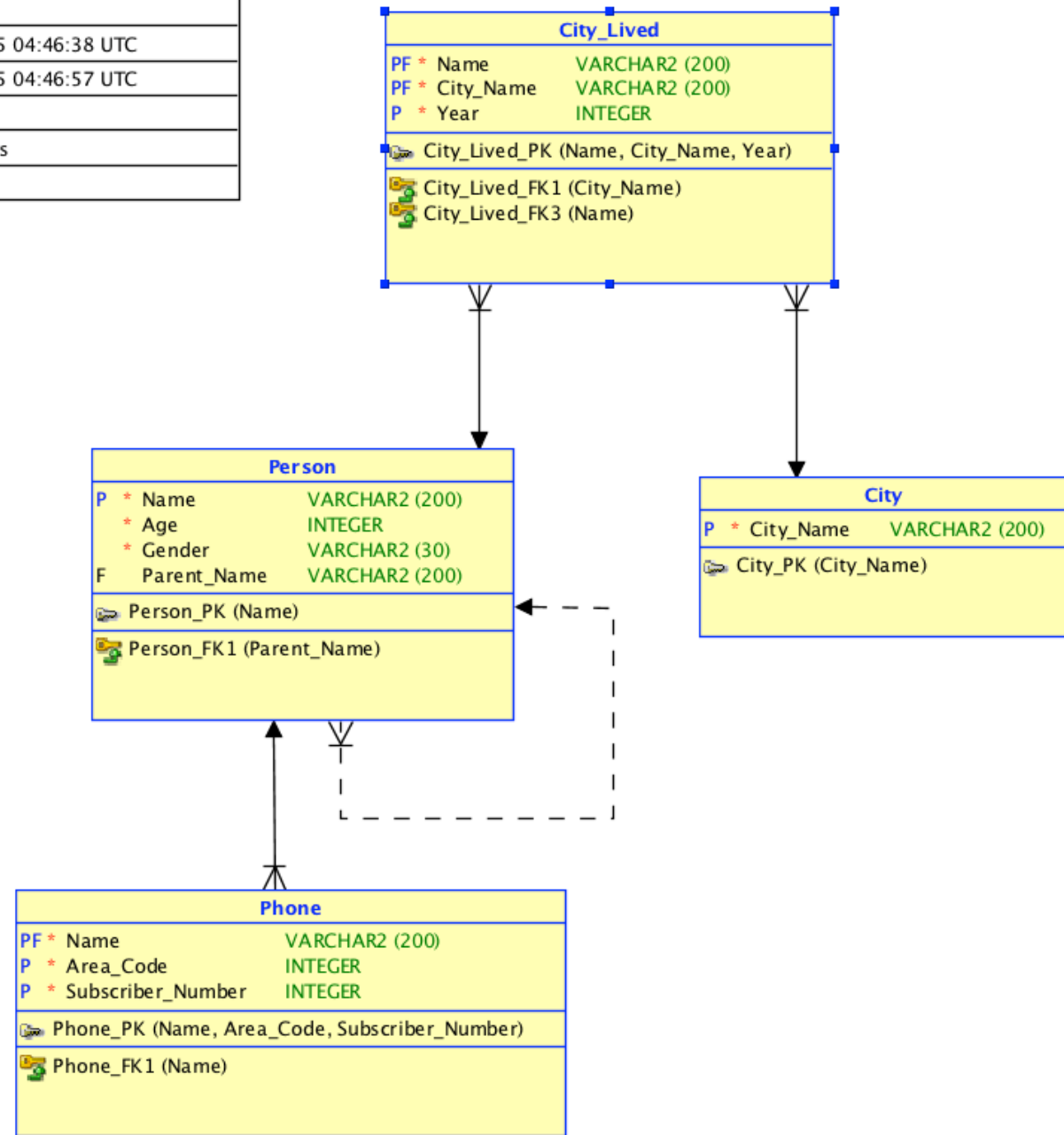


```
{  
  "cityName": "Portland",  
  "yearsLived": [ "1993", "1998", "2003", "2005" ]  
},
```

JSON as 3NF - Schema Model

- Can handle some JSON schema changes
 - Kids get a phone!
 - Kids move out!
- Extensions
 - More details on City
 - Add columns
 - More details on Children
 - Add columns or a dependent table

Diagram:	Physical 3NF (2)
Author:	kgraziano
Created on:	2018-02-05 04:46:38 UTC
Modified on:	2018-02-05 04:46:57 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	JSON 3NF



Data Vault Style

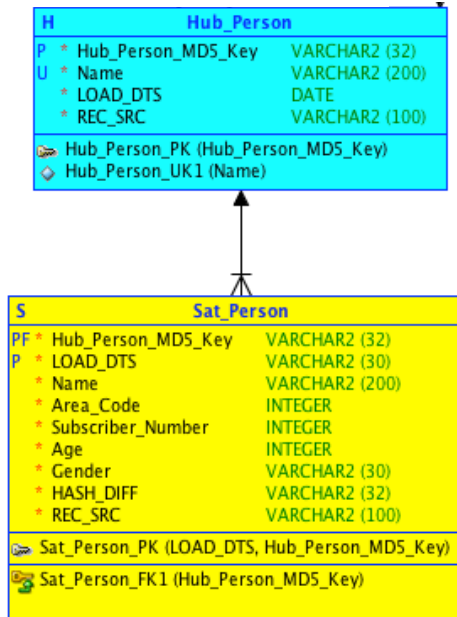
JSON as Data Vault

Diagram:	Data Vault 2
Author:	kgraziano
Created on:	2018-02-05 05:09:24 UTC
Modified on:	2018-02-05 05:09:24 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	JSON Data Vault

Hub_Person	
P	* Hub_Person_MDS_Key VARCHAR2 (32)
U	* Name VARCHAR2 (200)
	* LOAD_DTS DATE
	* REC_SRC VARCHAR2 (100)
	Hub_Person_PK (Hub_Person_MDS_Key)
	Hub_Person_UK1 (Name)

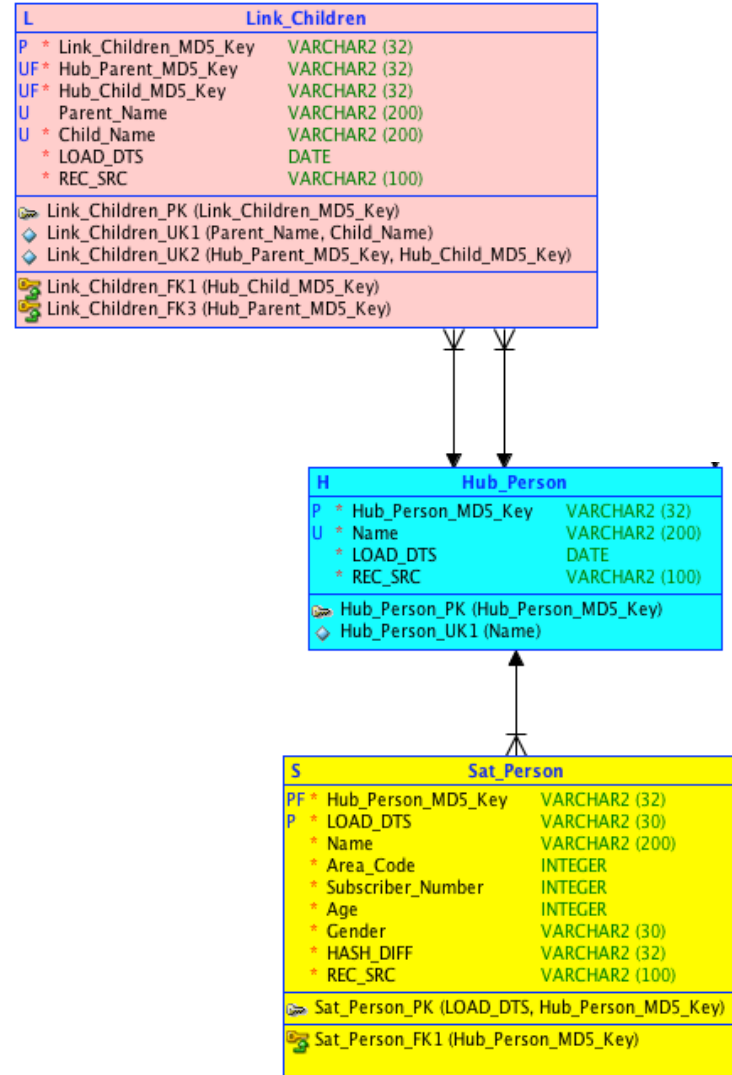
JSON as Data Vault

Diagram:	Data Vault 2
Author:	kgraziano
Created on:	2018-02-05 05:09:24 UTC
Modified on:	2018-02-05 05:09:24 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	JSON Data Vault



JSON as Data Vault

Diagram:	Data Vault 2
Author:	kgraziano
Created on:	2018-02-05 05:09:24 UTC
Modified on:	2018-02-05 05:09:24 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	JSON Data Vault



JSON as Data Vault

Diagram:	Data Vault 2
Author:	kgraziano
Created on:	2018-02-05 05:09:24 UTC
Modified on:	2018-02-05 05:09:24 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	JSON Data Vault

L Link_Children	
P *	Link_Children_MD5_Key VARCHAR2 (32)
UF *	Hub_Parent_MD5_Key VARCHAR2 (32)
UF *	Hub_Child_MD5_Key VARCHAR2 (32)
U	Parent_Name VARCHAR2 (200)
U	Child_Name VARCHAR2 (200)
*	LOAD_DTS DATE
*	REC_SRC VARCHAR2 (100)
Link_Children_PK (Link_Children_MD5_Key)	
Link_Children_UK1 (Parent_Name, Child_Name)	
Link_Children_UK2 (Hub_Parent_MD5_Key, Hub_Child_MD5_Key)	
Link_Children_FK1 (Hub_Child_MD5_Key)	
Link_Children_FK3 (Hub_Parent_MD5_Key)	

H Hub_Person	
P *	Hub_Person_MD5_Key VARCHAR2 (32)
U *	Name VARCHAR2 (200)
*	LOAD_DTS DATE
*	REC_SRC VARCHAR2 (100)
Hub_Person_PK (Hub_Person_MD5_Key)	
Hub_Person_UK1 (Name)	

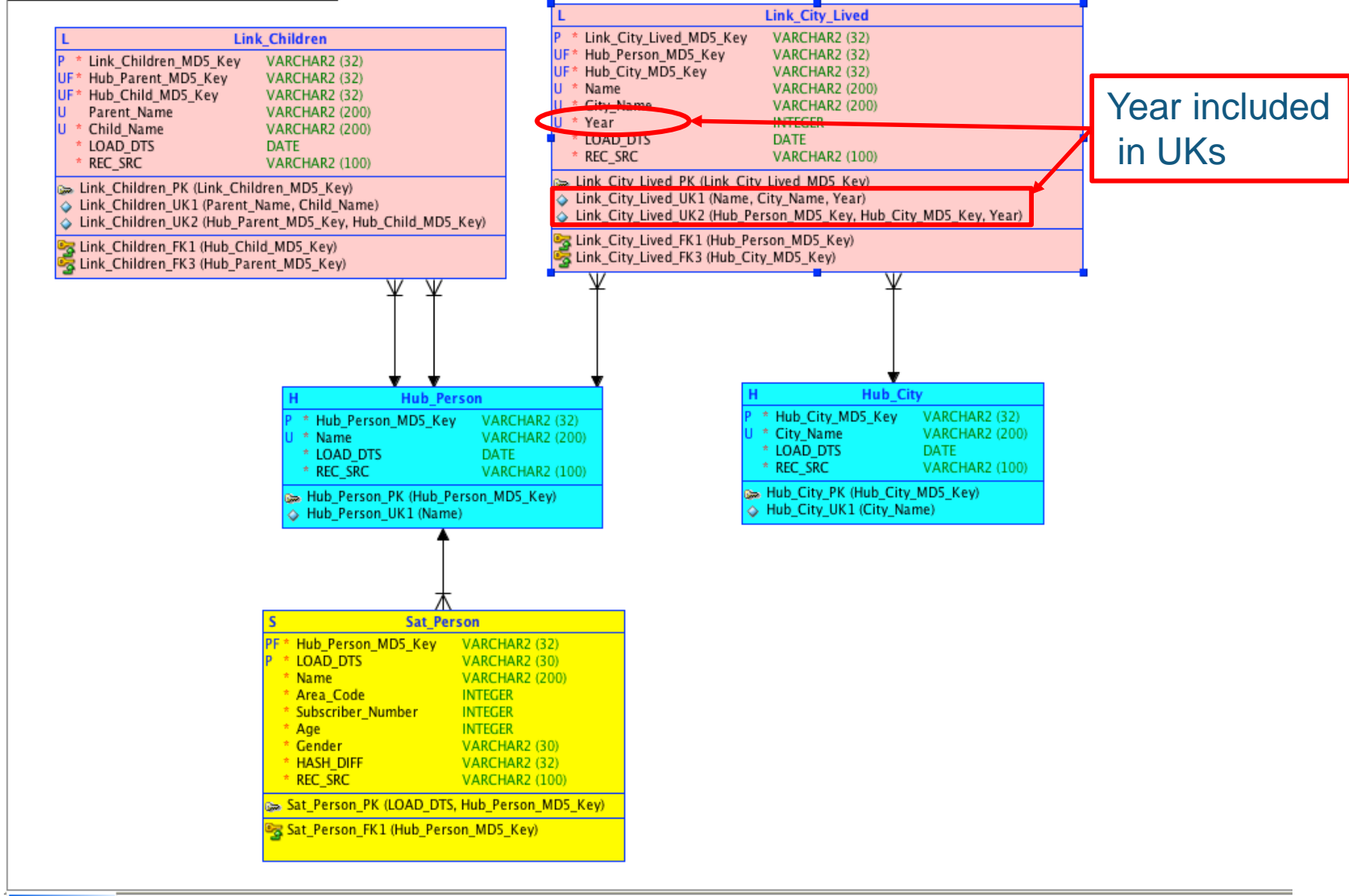
H Hub_City	
P *	Hub_City_MD5_Key VARCHAR2 (32)
U *	City_Name VARCHAR2 (200)
*	LOAD_DTS DATE
*	REC_SRC VARCHAR2 (100)
Hub_City_PK (Hub_City_MD5_Key)	
Hub_City_UK1 (City_Name)	

S Sat_Person	
PF *	Hub_Person_MD5_Key VARCHAR2 (32)
P *	LOAD_DTS VARCHAR2 (30)
*	Name VARCHAR2 (200)
*	Area_Code INTEGER
*	Subscriber_Number INTEGER
*	Age INTEGER
*	Gender VARCHAR2 (30)
*	HASH_DIFF VARCHAR2 (32)
*	REC_SRC VARCHAR2 (100)
Sat_Person_PK (LOAD_DTS, Hub_Person_MD5_Key)	
Sat_Person_FK1 (Hub_Person_MD5_Key)	

JSON as Data Vault

- Can handle some JSON schema changes
 - Two parents, same kids
 - Kids get a phone!
 - Kids move out!
- Easy Extensions
 - More details on City
 - Add a Sat
 - Add Link(s)
 - More details on Children
 - Add a Sat on Link

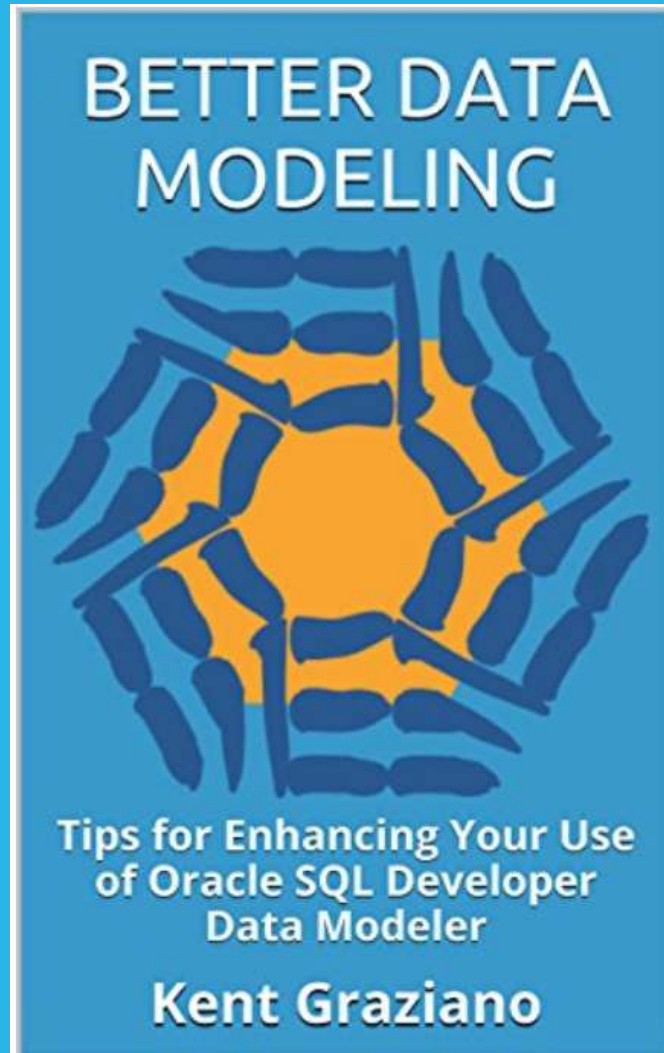
Diagram:	Data Vault 2
Author:	kgraziano
Created on:	2018-02-05 05:09:24 UTC
Modified on:	2018-02-05 05:09:24 UTC
Modified by:	kgraziano
Design:	JSON Models
Model:	JSON Data Vault



Conclusion

- We still need data models and data modelers
- Schema-on-Read does not mean there is no model
- To READ the data we must understand the SCHEMA
- In the DB world that means we need a model
 - Some model types can be easily extended for JSON changes
- Once the schema is understood
 - Can be expressed as any type of model
 - 3NF
 - Data Vault
 - Star
 - Denormalized
 - Object model
 - Etc.

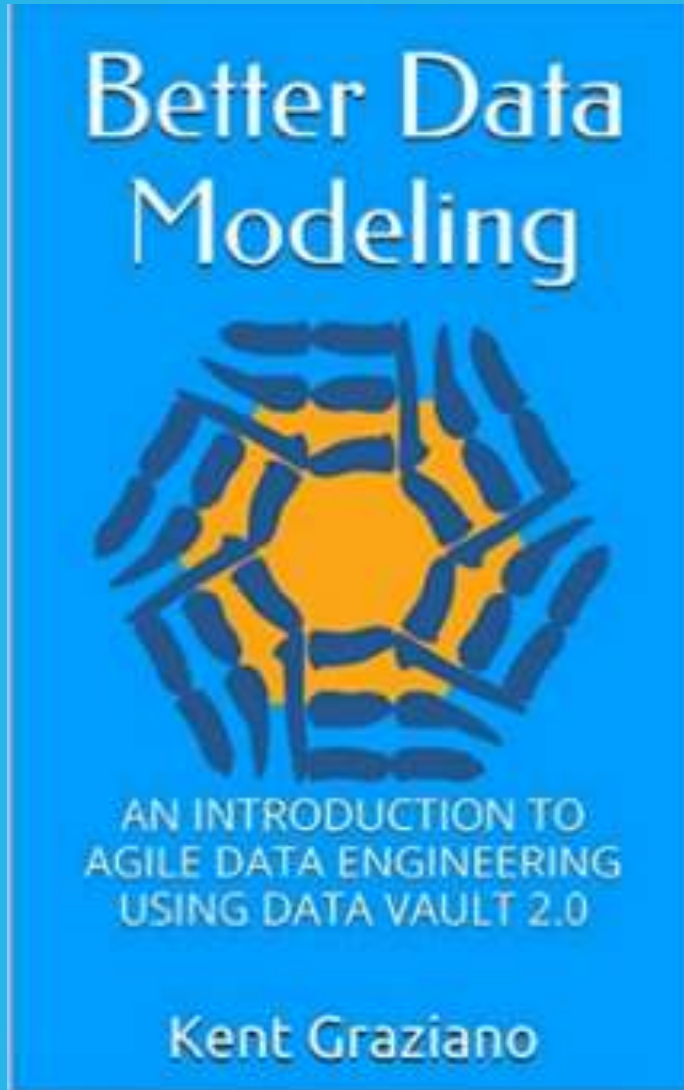
SHAMELESS PLUG:



**Available on
Amazon.com**

[https://www.amazon.com/
Better-Data-Modeling-
Enhancing-Developer-
ebook/dp/B00UK75LYI/](https://www.amazon.com/Better-Data-Modeling-Enhancing-Developer-ebook/dp/B00UK75LYI/)

SHAMELESS PLUG:



**Available on
Amazon.com**

**[http://www.amazon.com
/Better-Data-Modeling-
Introduction-
Engineering-
ebook/dp/B018BREV1C/](http://www.amazon.com/Better-Data-Modeling-Introduction-Engineering-ebook/dp/B018BREV1C/)**

Discover the performance, concurrency, and simplicity of Snowflake

As easy as 1-2-3!

- 01 Visit Snowflake.com
- 02 Click “Try for Free”
- 03 Sign up & register

Snowflake is the only data warehouse built for the cloud. You can automatically scale compute up, out, or down—independent of storage. Plus, you have the power of a complete SQL database, with zero management, that can grow with you to support all of your data and all of your users. With Snowflake On Demand™, pay only for what you use.

Sign up and receive **\$400** worth of free usage for 30 days!



Contact Information

Kent Graziano
Snowflake Computing
Kent.graziano@snowflake.com
On Twitter @KentGraziano

More info at
<http://snowflake.com>

Visit my blog at
<http://kentgraziano.com>





THANK YOU

