Data Warehousing in the Real World

Kent Graziano, Snowflake Computing
(Virtual) Keith Hoyle, McKesson Specialty Health
Agenda

- Bios
- Back story
- Standard DV Architecture
- Evolution to Gepetto
- How we use MD5 Hashes
- Planned Schema Architecture
- Final Schema Architecture
- Advantages & Challenges
My Bio

- **Senior Technical Evangelist**, Snowflake Computing
- Oracle ACE Director (BI/DW)
- Certified Data Vault Master and DV 2.0 Practitioner (CDVP2)
- Data Modeling, Data Architecture and Data Warehouse Specialist
- 30+ years in IT
- 25+ years of Oracle-related work
- 20+ years of data warehousing experience
- Former-Member: Boulder BI Brain Trust ([http://www.boulderbibraintrust.org/](http://www.boulderbibraintrust.org/))
- Author & Co-Author of a bunch of books
- Blogger: *The Data Warrior*
- Past-President of Oracle Development Tools User Group and Rocky Mountain Oracle User Group
Snowflake Computing is…
- …a Silicon Valley innovator
- …built a new SQL data warehouse in the cloud
- …with broad customer adoption

The Snowflake Elastic Data Warehouse is …
- …All-new, SQL compliant
  - No legacy code
- …Designed for the elastic cloud
- …Delivered as a service
  - Nothing to manage
Bio (Keith)

- Sr. Manager, Enterprise Data Architecture (McKesson Specialty Health)
- 25+ years in IT
- 8+ years in Genetic Engineering / Biochemistry in Pharmaceutical industry
- Completed multiple successful EDW efforts with large companies (Dell, HP, AMD, Aflac, Amgen, Glaxo-SmithKline, etc.)
- Consulted through large firms catering to big pharma / biotech / medical industry
Back story

- Client: McKesson Specialty Health (formerly US Oncology)
  - Division of McKesson (Fortune 500 - #5)
- Building a new Electronic Health Records (EHR) system
  - IKnowMed Generation 2 (G2)
- Existing DW on G1 – not good, not flexible
  - Pure Kimball – transient stage area with quasi-star schema model
  - Can’t handle multiple sources
  - Already issues loading and meeting SLA
Back story

- Want to build new DW
  - Flexible, scalable, etc.
- And want to use agile approach
- Sounds like Data Vault?
  - Contracted Kent to help
  - Hired Keith to be the internal lead
Standard Data Vault Model

- **Hub**: List of UNIQUE business keys.
- **Link**: List of UNIQUE
- **Satellite**: Historical descriptive data.

** Dashed Line is a possible New Relationship
Back story

- Management convinced that DV was too hard, too many layers, would take too long
  - Politics!

- So starting point – Type 2 style persistent stage area
  - Start loading ASAP
  - Never lose any changes
  - Good!
### Type 2 Stage Table

<table>
<thead>
<tr>
<th>S</th>
<th><strong>COMM_STG.G2_PRACTICE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>* PRIM_KEY</td>
</tr>
<tr>
<td>U</td>
<td>ID</td>
</tr>
<tr>
<td></td>
<td>CREATEDBY</td>
</tr>
<tr>
<td></td>
<td>CREATEDDATE</td>
</tr>
<tr>
<td></td>
<td>MODIFIEDBY</td>
</tr>
<tr>
<td></td>
<td>MODIFIEDDATE</td>
</tr>
<tr>
<td></td>
<td>MSCSACCOUNTNUMBER</td>
</tr>
<tr>
<td></td>
<td>NAME</td>
</tr>
<tr>
<td></td>
<td>ADDRESS1</td>
</tr>
<tr>
<td></td>
<td>ADDRESS2</td>
</tr>
<tr>
<td></td>
<td>CITY</td>
</tr>
<tr>
<td></td>
<td>CONTACTENMAIL</td>
</tr>
<tr>
<td></td>
<td>CONTACTNAME</td>
</tr>
<tr>
<td></td>
<td>FAXNUMBER</td>
</tr>
<tr>
<td></td>
<td>PHONENUMBER</td>
</tr>
<tr>
<td></td>
<td>STATE</td>
</tr>
<tr>
<td></td>
<td>TAXID</td>
</tr>
<tr>
<td></td>
<td>ZIPCODE</td>
</tr>
<tr>
<td></td>
<td>ELECTRONICFAXNUMBER</td>
</tr>
<tr>
<td></td>
<td>EXTENSION</td>
</tr>
<tr>
<td></td>
<td>FAXID</td>
</tr>
<tr>
<td></td>
<td>PHARMACYCONTACTEXTENSION</td>
</tr>
<tr>
<td></td>
<td>PHARMACYCONTACTPHONENUMBER</td>
</tr>
<tr>
<td></td>
<td>PUBLICID</td>
</tr>
<tr>
<td></td>
<td>TIMEZONE</td>
</tr>
<tr>
<td></td>
<td>PATIENTPORTALACTIVE</td>
</tr>
<tr>
<td></td>
<td>CDC_KEY</td>
</tr>
<tr>
<td></td>
<td>CDC_ATTR</td>
</tr>
<tr>
<td></td>
<td>NAT_KEY</td>
</tr>
<tr>
<td></td>
<td>REC_SRC</td>
</tr>
<tr>
<td></td>
<td>REC_SRC_TBL</td>
</tr>
<tr>
<td></td>
<td>LOAD_DTS</td>
</tr>
<tr>
<td></td>
<td>SRC_DEL_FLG</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th><strong>COMM_STG.KDW_ORG</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>* PRIM_KEY</td>
</tr>
<tr>
<td>U</td>
<td>ORGID</td>
</tr>
<tr>
<td></td>
<td>ORG_NAME</td>
</tr>
<tr>
<td></td>
<td>ORG_ABBR</td>
</tr>
<tr>
<td></td>
<td>ORGIND</td>
</tr>
<tr>
<td></td>
<td>CUSTOMERIND</td>
</tr>
<tr>
<td></td>
<td>LOCATIONIND</td>
</tr>
<tr>
<td>U</td>
<td>PARENTORGID</td>
</tr>
<tr>
<td></td>
<td>PATSRCHLEVELORGID</td>
</tr>
<tr>
<td></td>
<td>TIMEZONE</td>
</tr>
<tr>
<td></td>
<td>EXTRACT_DTE</td>
</tr>
<tr>
<td></td>
<td>LOAD_DTS</td>
</tr>
<tr>
<td>F</td>
<td>* CDC_KEY</td>
</tr>
<tr>
<td></td>
<td>CDC_ATTR</td>
</tr>
<tr>
<td></td>
<td>NAT_KEY</td>
</tr>
<tr>
<td></td>
<td>REC_SRC</td>
</tr>
<tr>
<td></td>
<td>REC_SRC_TBL</td>
</tr>
<tr>
<td></td>
<td>SRC_DEL_FLG</td>
</tr>
<tr>
<td></td>
<td>EXPR_DTS</td>
</tr>
</tbody>
</table>

- **G2_PRACTICE_PK** (PRIM_KEY)
- **G2_PRACTICE_NK** (NAME)
- **G2_PRACTICE_CODE** (ID)
- **G2_PRACTICE_IDIX1** (PRIM_KEY)
Evolution of Gepetto

- Initial “marts” were just views off the stage tables
  - Joins in Business Objects
  - Worked fine for 1 source (G2)
- But what happens when you add another source?
  - Explosion of mappings from stage to presentation
  - Mapping logic in ETL or complex views
- Need: a persistent integration layer
  - Based on natural **business keys**!
  - But don’t say “data vault”
Persistent Integration Layer Advantage

No PIL (M x N)

With a PIL (M + N)

Stage 1
Stage 2
Stage 3
Stage 4
Stage 5
Stage 6

Dim 1
Dim 2
Dim 3
Fact 1
Fact 2
Fact 3

Persistent Integration Layer
INTRODUCING GEPETTO!
Development Approach

- While source data was being staged:
  - Ran JAD sessions determining business information model for integration
  - Standardization routines developed
    - Full featured, configurable Calendar Dimension
    - Standardized plumbing columns and CDC logic
    - Consistent means of MD5 hashing
  - Persistent integration layer developed
  - Prototyped merging data from multiple sources into conformed hybrid SCD-1 / 2 dimensions
  - Devised highly-normalized ‘Gepetto’ variant of DV 2.0 / Anchor methods
Gepetto Architecture

- Very “vault-like”
- Keys are all MD5 based hash types
- 3 layers
  - Stage
  - Integration
  - Presentation
- Integration
  - Domains (business key driven like Hubs)
  - Relaters (basically Links)
  - Key Map table – joins D & R to stage tables
    - Stage tables act like Satellites
Domain and Key Map Tables

**COMN_INTG.D_LOC**

- **D_LOC_KEY**: VARCHAR2 (32)
- **LOC_NM**: VARCHAR2 (80)
- **LOC_STREET_ADDR_1**: VARCHAR2 (200)
- **LOC_STREET_ADDR_2**: VARCHAR2 (200)
- **LOC_STREET_ADDR_3**: VARCHAR2 (200)
- **LOC_CITY_NM**: VARCHAR2 (80)
- **LOC_ST_CD**: VARCHAR2 (10)
- **LOC_ZIP_CD**: VARCHAR2 (10)
- **REC_SRC**: VARCHAR2 (80)
- **REC_SRC_TBL**: VARCHAR2 (80)
- **LOAD_DTS**: TIMESTAMP

**COMN_INTG.KM_LOC**

- **D_LOC_KEY**: VARCHAR2 (32)
- **CDC_KEY**: VARCHAR2 (32)
- **REC_SRC**: VARCHAR2 (80)
- **REC_SRC_TBL**: VARCHAR2 (80)
- **LOAD_DTS**: TIMESTAMP
- **EXPR_DTS**: TIMESTAMP

**Key Maps**

- **COMN_INTG.D_LOC**
  - D_LOC_PK (D_LOC_KEY)
  - D_LOC_BK (LOC_NM, LOC_STREET_ADDR_1, LOC_STREET_ADDR_2, LOC_STREET_ADDR_3, LOC_CITY_NM, LOC_ST_CD, LOC_ZIP_CD)

- **COMN_INTG.KM_LOC**
  - KM_LOC_PK (D_LOC_KEY, CDC_KEY, REC_SRC, REC_SRC_TBL, LOAD_DTS)
  - KM_LOC_CDCKI (CDC_KEY)
MD5 Keys

- Concatenate source data fields and hash to create MD5 keys

Concatenation Rules
- Joins are performed against these keys so standards and consistency are vital
- Use a delimiter when concatenating
- Convert numbers and dates/times to string
- Consider trimming/upper casing values in BUS_KEYS

MD5 Key Types
- **PRIM_KEY (STG):**
  - All source fields (in table order) + LOAD_DTS
  - Uniquely ID’s all records with DW
  - Can serve as an SCD-2 key in virtual Dim’s/ Facts
MD5 Keys

- **CDC_KEY (STG / INTG):**
  - Source field(s) (in table order) used by SOR to ID data rows uniquely for change data capture purposes
  - Same as MD5Key in DV 2.0

- **CDC_ATTR (STG):**
  - All non-CDC_KEY source field(s) (in table order) to track changed for change data capture purposes
  - Same as MD5DIFF in DV 2.0

- **NAT_KEY (STG):**
  - Source field(s) (in table order) from a single SOR table used to logically ID data rows uniquely
  - Table “natural” key is not always a true business key
MD5 Keys

- \([D\_\text{XXX\_KEY} / R\_\text{XXX\_KEY}]\) (INTG):
  - Hash of real business key columns
  - Source field(s) (in table order) used to logically ID data rows uniquely
    - Joins may be required because of the nature of the stage tables
  - Same as HUB and LINK keys in DV 2.0
  - Can serve as an Type 1 SCD key in virtual Dim’s/ Facts
    - That is another talk!
What does it look like?

- Encode using standard MD5 hash function
  - rawtohex(sys.utl_raw.cast_to_raw(
    dbms_obfuscation_toolkit.md5 (input_string => ...)
  )

- Need to minimize chance of duplicates
  - 12||3||45 and 1||2||345 hash to same value
  - Need a separator between each
  - Also handles case of null values
  - Example: Col1||’^’||Col2||’^’||Col3
Other considerations

- To generate most consistent string: standardize!
- Convert data types
- If 'NUMBER', 'NCHAR', 'NVARCHAR2', 'NVARCHAR', 'NCHAR'
  - THEN 'TO_CHAR(' || column_name || ')
- If 'RAW'
  - THEN 'ENC_BASE64(' || column_name || ')
- If 'DATE'
  - THEN 'TO_CHAR(' || column_name || ', "YYYY-MM-DD")'
- If LIKE 'TIME%'
  - THEN 'TO_CHAR(' || column_name || ', "YYYY-MM-DD HH24:MI:SS")'
(UPPER(TRIM(T1.GENERICNAME))
||'^'||
UPPER(TRIM(
    TO_CHAR(T1.MED_STRNG_AMT))
||'^'||
UPPER(TRIM(T1.UOM_CD))
||'^'||
UPPER(TRIM(T1.MED_FORM_NM))
||'^')
So what?

- MD5 hash is consistent cross-platform
- Changes multi-column compares to a single column
- All compares take the same time during load process
- Can use with any DW architecture that requires change detections
- Virtually no limit
  - Think Big Data/Hadoop/NoSQL
- Can generate the input string automatically
  - But that is another talk!
ARCHITECTURE OVERVIEW
Original Schema Architecture

Source(s) of Record

| HI       | FDW / PMS | G2     | EDW V1 | KDW | Lynx | SFDC |

Insert 1X only

COMN Integration

*Enterprise business key model with key mapping pointers to COMN_STG data>*

COMN Stage

*Full copies of source data structures with additional plumbing fields to facilitate capturing subsequent data changes over time>*

JIT Transformation

*Virtual v. Physical>*

Data Marts

MSH EDW

Reporting

TBLU

Data Reporting

BOBJ

Data Reporting

Web

Data Reporting
Hoped for Schema Architecture (Parallel)

Source(s) of Record

- HI
- FDW / PMS
- G2
- EDW V1
- KDW
- Lynx
- SFDC

MSH EDW

- HI Stage
- FIN Stage
- HI Presentation
- FIN Presentation
- COMN Integration
- COMN Stage
- COMN Validation
- COMN Presentation

BOBJ / BI / Reporting

- HI
- FIN
- CLIN
- MKTG

KDW Lite
- MU
- CI SAS Routines
Challenges

- Must have a solid enterprise logical model
  - With real business keys!
- Mapping disparate sources to the Integration layer is hard!
  - Must understand the semantic meaning of the source columns
  - Must know the enterprise model to see where it fits
  - Must know how to handle bad and missing business key data
    - Means you must have good business rules too!
- Dimensional modelers have a hard time with doing these mappings.
  - Using views in Presentation layer mitigates this by displaying in star manner to BI layer
Advantages

- Can start staging immediately
  - With history!
- Clear line of sight to source
  - Unambiguous audit trail
- Can adapt, recovering from incorrect business rules
  - Stage data is in original source format, with history
Old Way vs New Way

Cowpath

Highway

Which way will you follow?
SHAMELESS PLUG:

Available on Amazon.com

Super Charge Your Data Warehouse

Available on Amazon.com
Soft Cover or Kindle Format

Now also available in PDF at LearnDataVault.com
New DV 2.0 Book (includes more details on MD5)

Available on Amazon: http://www.amazon.com/Building-Scalable-Data-Warehouse-Vault/dp/0128025107/
Questions?
Contact Information

Kent Graziano
Snowflake Computing

Kent.graziano@snowflake.net
@KentGraziano
http://kentgraziano.com

The Data Warrior
Changing the world, one data model at a time. How can I help you?
Contact Information

Keith Hoyle
Sr. Mgr., Enterprise Data Architecture
McKesson Specialty Health
Khoyle001@hotmail.com
Visit my blog at
http://khoyle001.wordpress.com