A deep dive into a Data Warehouse

Office of Institutional Effectiveness Humboldt State University

Data Warehouse:

Strategic Data Repository - SDR

Presenters:

Ward Headstrom, Data Scientist
ward.headstrom@humboldt.edu

Ronda Stemach, Data Administrator
ronda.stemach@humboldt.edu

Michael Le, Research Associate
michael.le@humboldt.edu

CAIR 2016

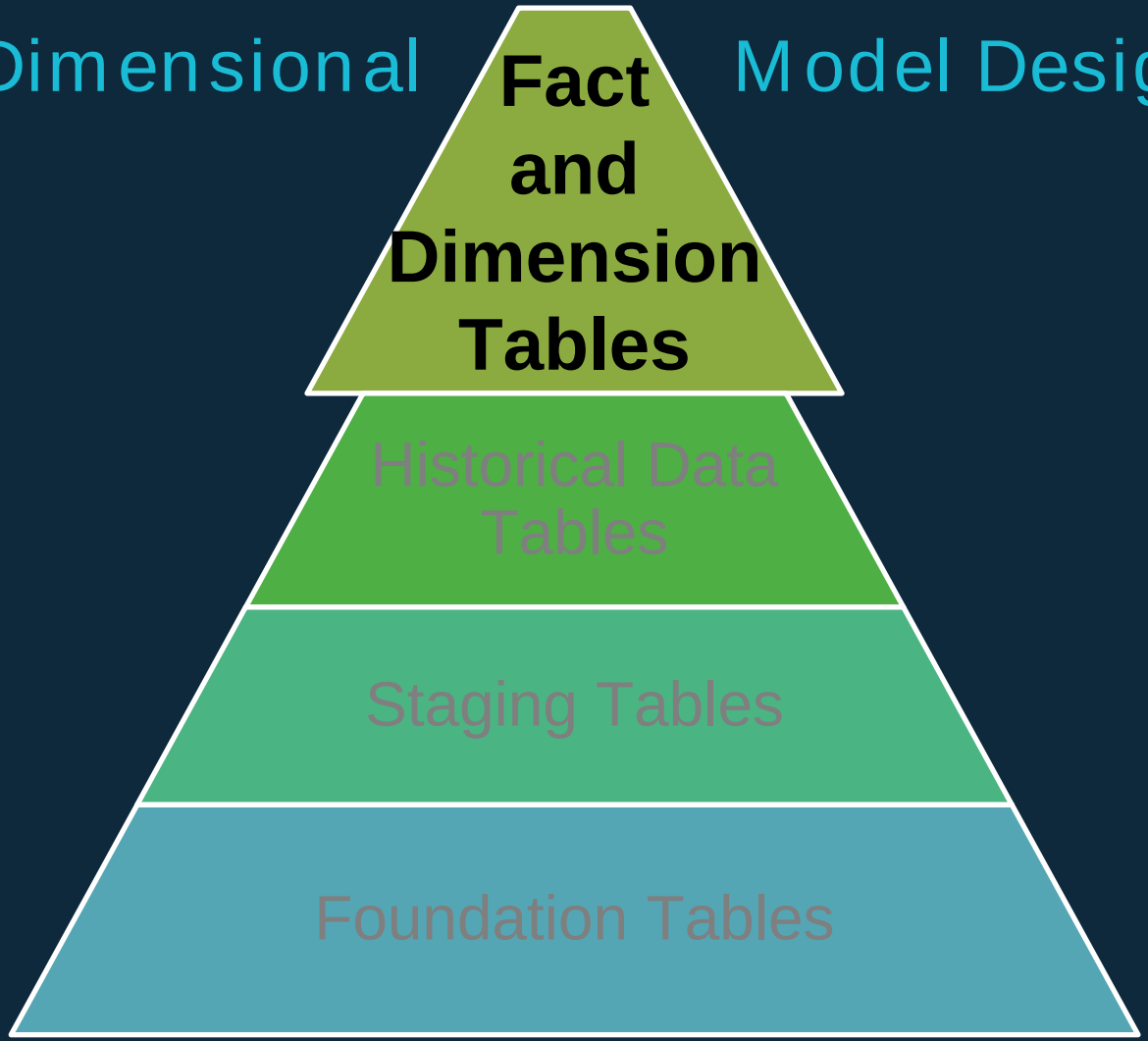
CAIR
2016

Tableau Dashboards

Dimensional Model Design

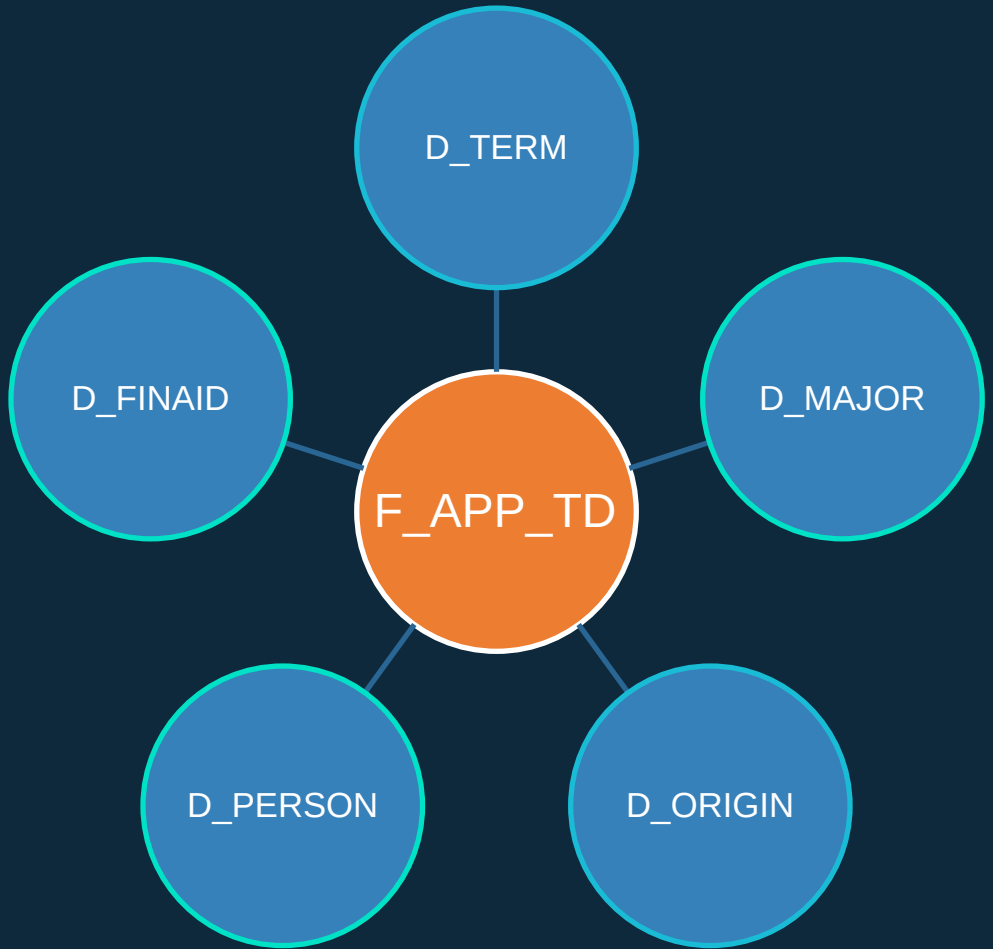
CAIR
2016

Dimensional Model Design



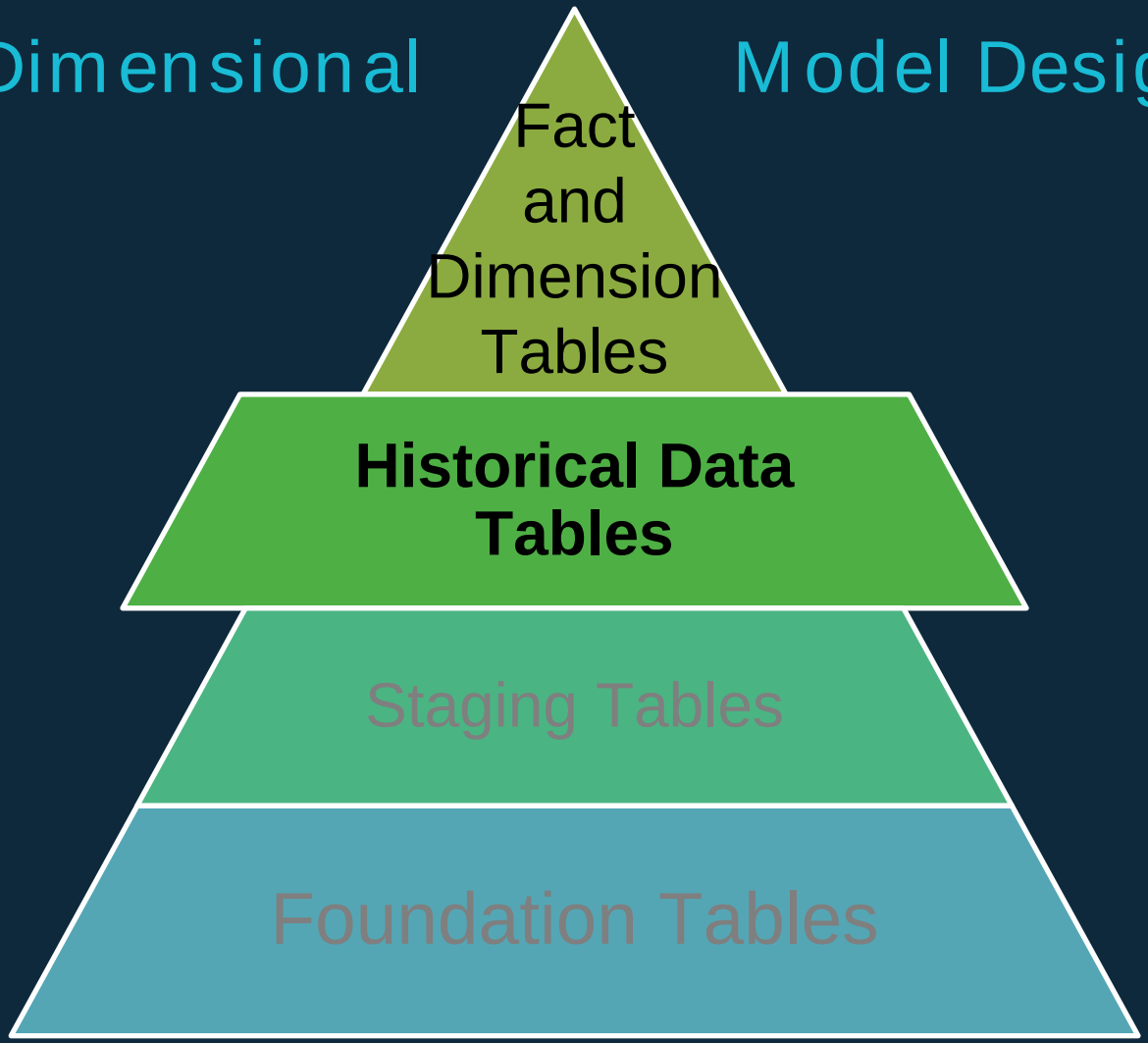
CAIR 2016

Application Fact and Dimensions

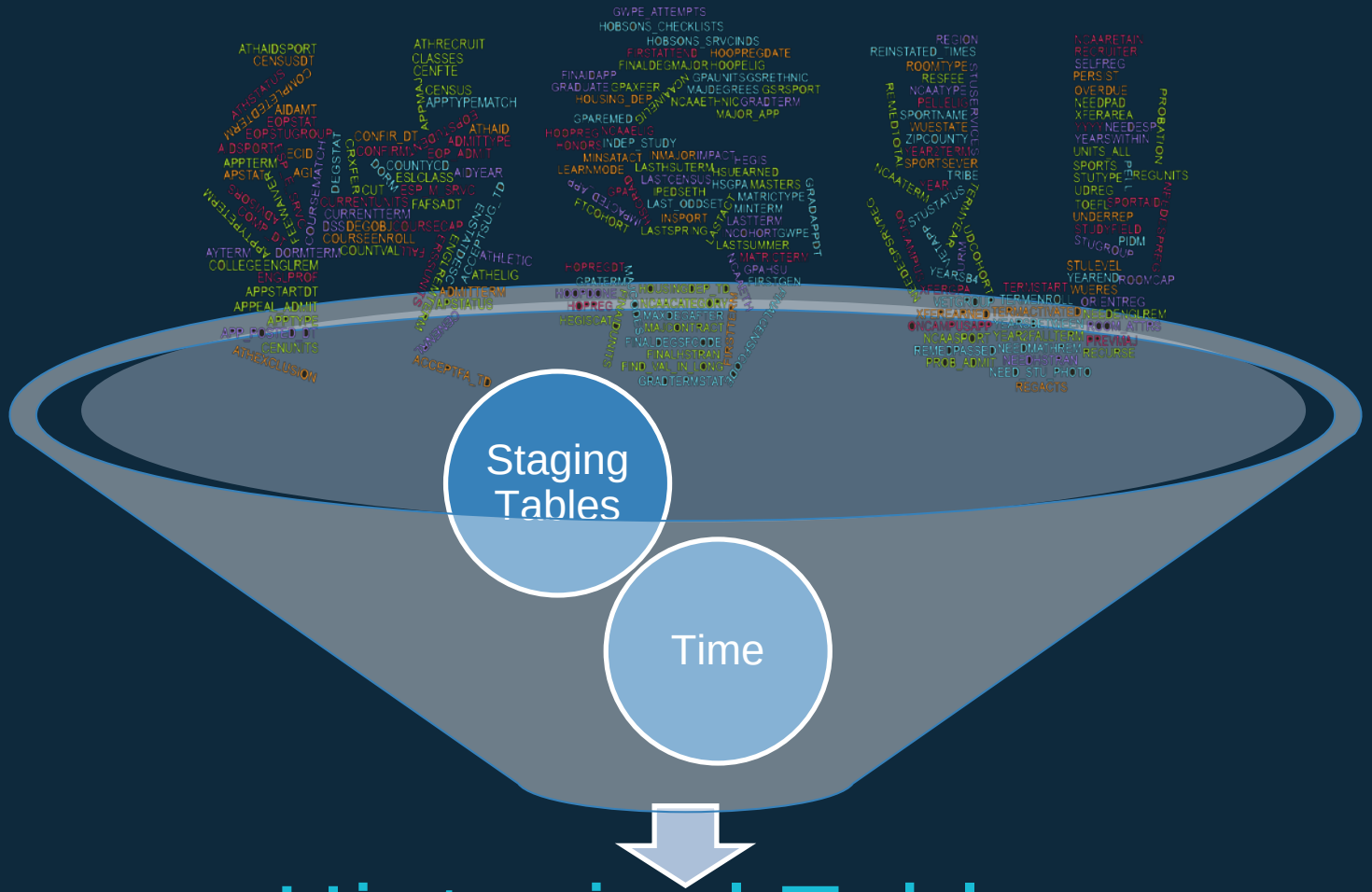


CAIR
2016

Dimensional Model Design



CAIR 2016



Historical Tables

CAIR 2016

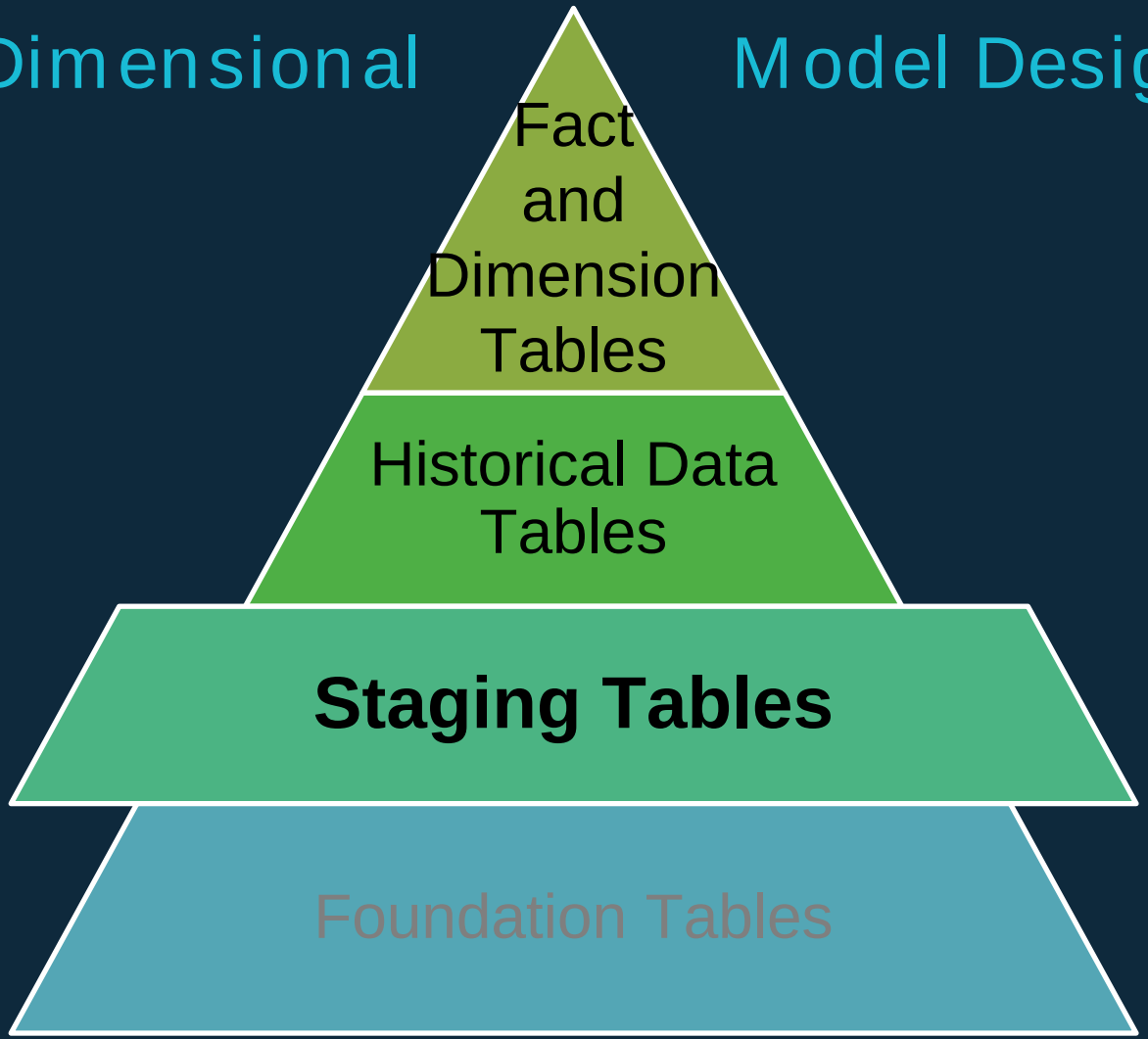
APP_CDC

	APP_KEY	TERM_KEY	PERSON_KEY	MAJORS_KEY	ORIGIN_KEY	FINAID_KEY	APP_DT	ADMIT_DT	CONFIRM_DT	EFFDT	ENDDT	LATEST	OBSOLETE	APP_SDR_KEY	HC_APP	HC_EX
1	2174013112174	2174	013112174	SW	C31032203	2018013112174	02-OCT-16	(null)	(null)	04-OCT-16	05-OCT-16	N	(null)	37345217	1	
2	2174013112174	2174	013112174	SW	C31032203	2018013112174	02-OCT-16	(null)	(null)	06-OCT-16	06-OCT-16	N	(null)	37345635	1	
3	2174013112174	2174	013112174	SW	C31032203	2018013112174	02-OCT-16	(null)	(null)	07-OCT-16	07-OCT-16	N	(null)	37346000	1	3.46
4	2174013112174	2174	013112174	SW	C31032203	2018013112174	02-OCT-16	(null)	(null)	08-OCT-16	13-OCT-16	N	(null)	37346569	1	1.63
5	2174013112174	2174	013112174	SW	C31032203	2018013112174	02-OCT-16	(null)	(null)	14-OCT-16	24-OCT-16	N	(null)	37347872	1	0.5443
6	2174013112174	2174	013112174	SW	C31032203	2018013112174	02-OCT-16	24-OCT-16	(null)	25-OCT-16	31-DEC-99	Y	(null)	37349911	1	0.3598

Other History Tables Include:

- Person
- Major
- Academic Program
- Class
- Application
- Enrollment
- Degree
- Financial Aid
- Athletic Participation

Dimensional Model Design



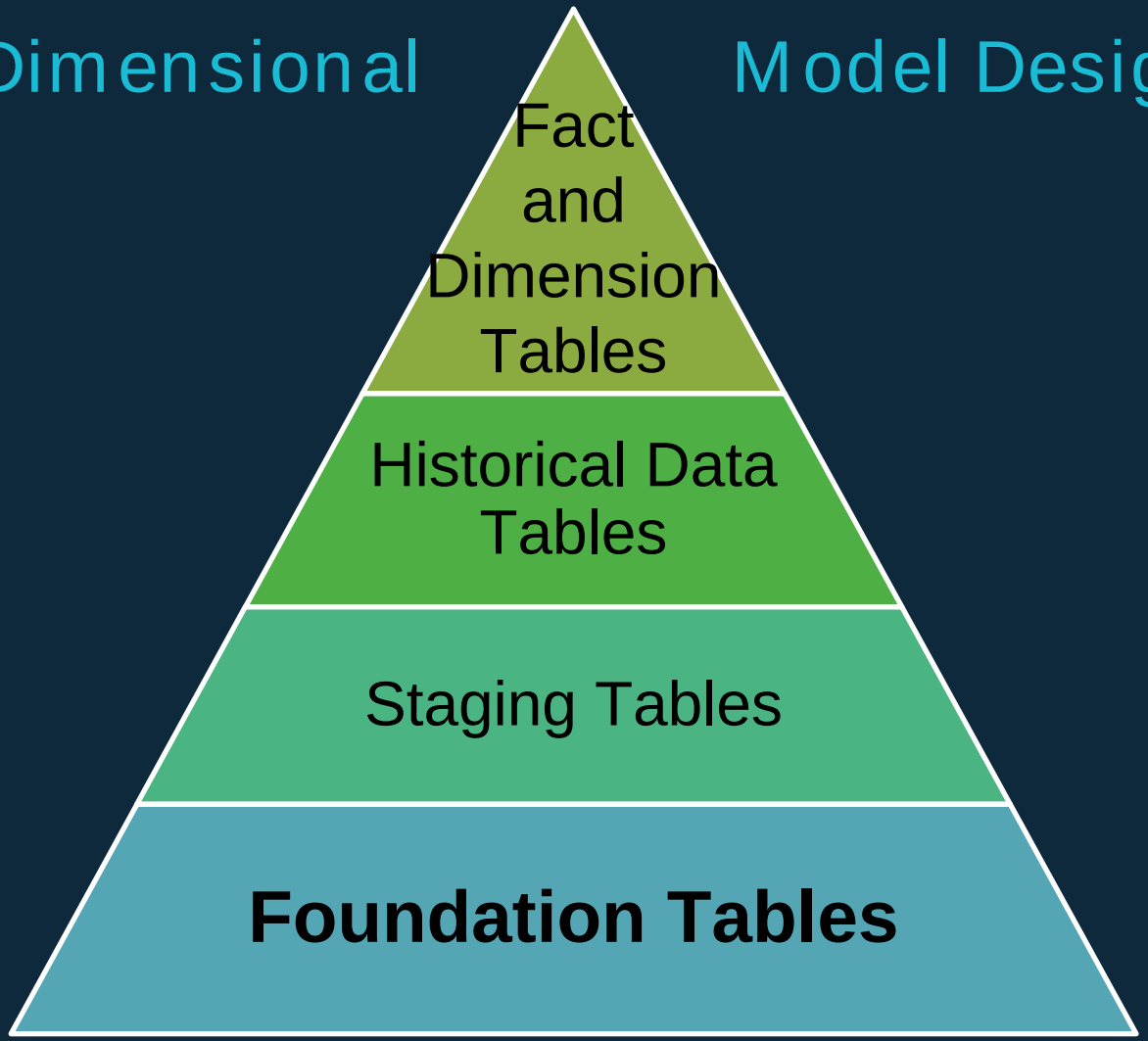
CAIR 2016

Staging Tables

- ps_csu_adm_excp_cd
- ps_adm_app_fee
- ps_residency_off
- ps_ext_acad_sum
- ps_adm_appl_eval
- ps_adm_appl_data
- ps_acad_prog
- ps_adm_appl_prog
- ps_adm_appl_plan
- ps_csu_adm_ap_data
- ps_ext_acad_data

CAIR 2016

Dimensional Model Design



Foundation Tables

County

	CSUCODE	COUNTYNAME	REGION	RECRUITER
1	01	ALAMEDA	3-SF Bay	SF
2	02	ALPINE	6-Central CA	Sac
3	03	AMADOR	6-Central CA	Sac
4	04	BUTTE	2-Northern CA	NE
5	05	CALAVERAS	6-Central CA	Central
6	06	COLUSA	2-Northern CA	NE
7	07	CONTRA COSTA	3-SF Bay	SF
8	08	DEL NORTE	1-Local	NW
9	09	EL DORADO	6-Central CA	Sac
10	10	FRESNO	6-Central CA	Central
11	11	GLENN	2-Northern CA	NE
12	12	HUMBOLDT	1-Local	NW

Compdates

	KEY	123 ...	STARTDATE	THISYEAR	LASTYEAR	DESCRIPT
1	580	2174	01-OCT-16	01-OCT-16	01-OCT-15	begin accepting applications
2	560	2172	01-AUG-16	01-AUG-16	01-AUG-15	begin accepting applications
3	490	2164	12-APR-16	11-APR-16	13-APR-15	start of web registration
4	550	2164	21-JUN-16	20-JUN-16	22-JUN-15	HOOP 2nd session
5	554	2164	09-AUG-16	08-AUG-16	10-AUG-15	HOOP 3rd session
6	543	2164	26-MAY-16	25-MAY-16	27-MAY-15	HOOP registration starts
7	541	2164	22-JUN-16	20-JUN-16	15-JUN-15	start of HOP
8	493	2164	19-DEC-16	19-DEC-16	20-DEC-15	last day of class
9	492	2164	15-SEP-16	22-SEP-16	23-SEP-15	Census
10	491	2164	25-AUG-16	25-AUG-16	26-AUG-15	first day of class
11	494	2164	01-OCT-15	01-OCT-15	01-OCT-14	begin accepting applications
12	505	2162	08-FEB-16	15-FEB-16	16-FEB-15	Census
13	506	2162	13-MAY-16	13-MAY-16	15-MAY-15	last day of class

ZipCode

	ZIP	COUNTYNAME	CITY	AREACODES	CSUCODE
4	92301	SAN BERNARDINO	ADELANTO	760	36
5	96006	MODOC	ADIN	530	25
6	91301	LOS ANGELES	AGOURA HILLS	818	19
7	91376	LOS ANGELES	AGOURA HILLS	805/818	19
8	92536	RIVERSIDE	AGUANGA	909	33
9	93601	MADERA	AHWAHNEE	559	20
10	94501	ALAMEDA	ALAMEDA	510	01
11	94502	ALAMEDA	ALAMEDA	510	01
12	94507	CONTRA COSTA	ALAMO	925	07
13	94706	ALAMEDA	ALBANY	510	01
14	95410	MENDOCINO	ALBION	707	23
15	95511	HUMBOLDT	ALDERPOINT	707	12
16	91841	LOS ANGELES	ALHAMBRA	626	19
17	91804	LOS ANGELES	ALHAMBRA	626	19
18	91803	LOS ANGELES	ALHAMBRA	323/626	19

Decision

	ACTION	REASON	ACTIVE	APPLIED	ADMITTED	PROVISIONAL	AUTOEVAL	CONFIRMED	DENIED	CANCELLED	WITHDRAWN	COMPLETE	REVIEW	EOP_REVIEW	XPAD	DESCRIPTION
1	ADMT	-	Y	Y	Y	N	N	(null)	N	N	(null)	(null)	N	N	(null)	Admit without reason
2	ADMT	ADRE	Y	Y	Y	N	N	(null)	N	N	(null)	(null)	N	N	(null)	Admitted-Reinstated
3	ADMT	ADMT	Y	Y	Y	N	N	(null)	N	N	(null)	(null)	N	N	(null)	Admitted Fully
4	ADMT	ADPB	Y	Y	Y	N	N	(null)	N	N	(null)	(null)	N	N	(null)	Admitted on Probation
5	ADRV	-	N	(null)	N	(null)	(null)	(null)	Y	(null)	(null)	(null)	N	N	(null)	Admission Revoked
6	ADRV	ADRO	N	(null)	N	(null)	(null)	(null)	Y	(null)	(null)	(null)	N	N	(null)	AdmisCancel-Did Not Clear Prov
7	ADRV	REVO	N	(null)	N	(null)	(null)	(null)	Y	(null)	(null)	(null)	N	N	(null)	Admis Revoked Stdt Req
8	ADRV	DEAT	N	(null)	(null)	(null)	(null)	(null)	(null)	Y	(null)	(null)	(null)	(null)	(null)	Death
9	APPL	-	Y	Y	N	N	N	N	N	N	N	N	N	N	N	Applied
10	APPL	APPD	Y	Y	N	N	N	N	N	N	N	N	N	N	N	Applied
11	COND	AUTO	Y	(null)	Y	Y	Y	(null)	N	(null)	(null)	(null)	(null)	(null)	(null)	Admitted Prov via Auto Eval
12	COND	ADPR	Y	(null)	Y	Y	N	(null)	N	N	(null)	(null)	(null)	(null)	(null)	Admitted Provisionally

NCAA Identifiers

ETL Errors & logs

Data Audits

CAIR 2016

Design Principles/ Practices

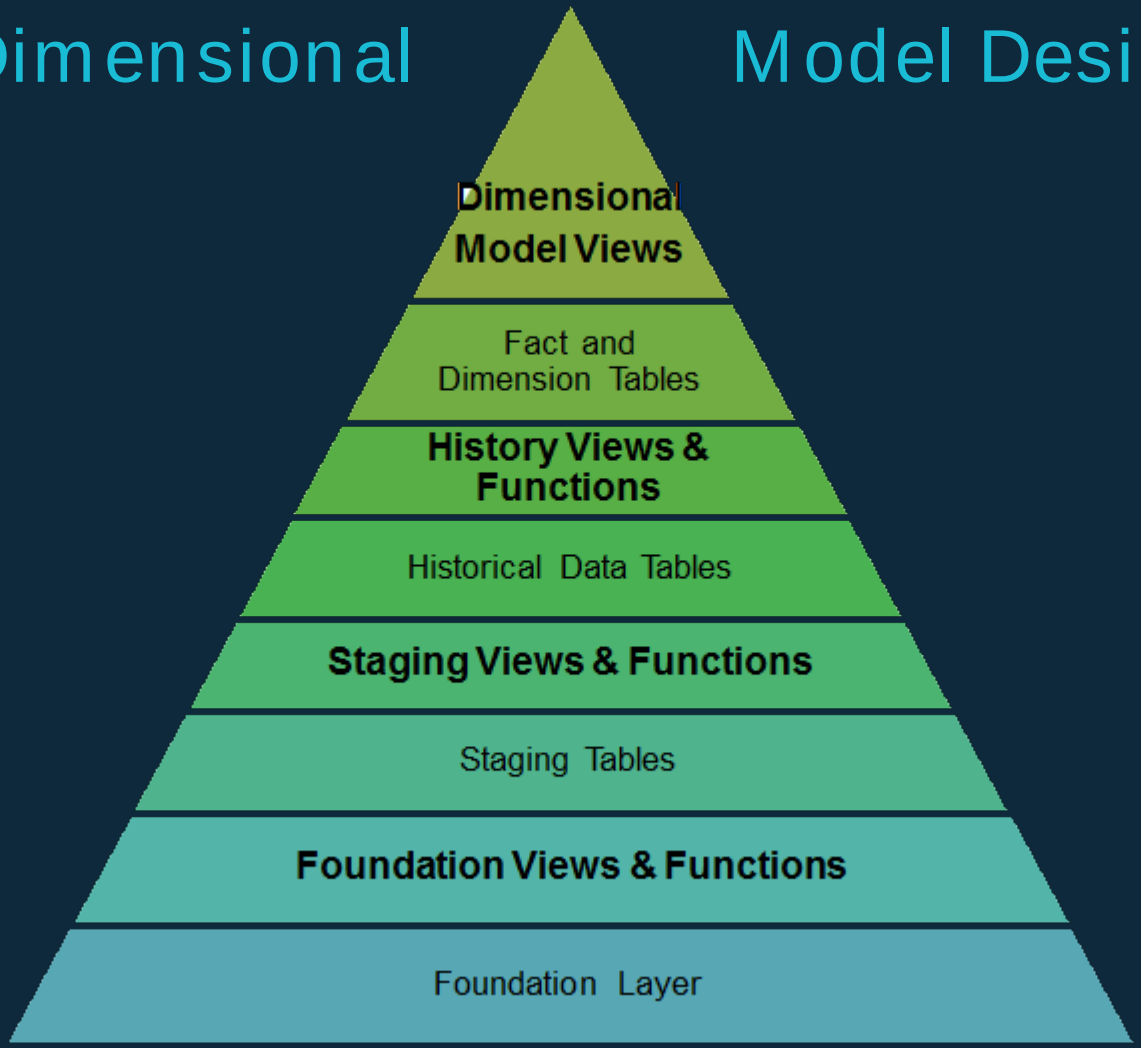
CAIR
2016

- Views and Functions
  - 90% of what we do is writing and testing SQL objects
    - Business practices built into views and functions (>1500)
    - Only refer to each table once
    - Layer views
- Build small SQL tools to eliminate repetition and increase reliability
  - download: copy a SIS table to local database and build index
  - effdt: create stage 0 and 1 views of a staging table
  - fieldsnotin: return fields in a view that are not in another view
  - cdc: capture new and changed data records in a history table
  - dmgen: “materialize” a history view and add indexes
  - cout: check out a script from RCS
  - cin: check in a script to RCS and mark as modified
  - refresh: reload all SQL objects that have been modified
  - rev: revalidate all SQL objects
  - fvcol: find all views which reference a particular column

ETL Process (nightly)

- Destructive rebuild of staging tables
  - Table-driven download of 220 tables, 90M records
    - Approximately one hour
- Nondestructive appends to history tables
  - Daily snapshot of “current values”
  - Append to about 20 tables
  - Approximately one hour
- Destructive build of dimensional tables
  - Rebuild about 35 fact tables and 25 dimension tables
  - Approximately two hours

Dimensional Model Design



CAIR 2016

Staging tables - download

Add table name to download table and run download script (or wait for it to run automatically at the beginning of the nightly ETL).

SA_DOWNLOAD

TABLENAME	NIGHTLY	DOWNLOAD_GROUP	DOWNLOAD_DT	DOWNLOAD_ROWS
ps_adm_appl_data	Y	app	11/10/2016 12:09:56 AM	325724
ps_adm_appl_plan	Y	app	11/10/2016 12:10:24 AM	1022492
ps_adm_appl_prog	Y	app	11/10/2016 12:10:46 AM	1018323
ps_csu_adm_ap_data	Y	app	11/10/2016 12:14:51 AM	325774
ps_csu_adm_app_fee	Y	app	11/10/2016 12:14:16 AM	146750
ps_hum_admconf_vw	Y	app	11/10/2016 12:26:07 AM	3134

Staging views - stage 0 and 1

Add aliases and handle effective-dating:

```
define table=ps_adm_appl_plan
define view=adm_appl_plan
define grain=emplid,acad_career,stdnt_car_nbr,adm_appl_nbr,appl_prog_nbr
define alias="emplid id,acad_career career,adm_appl_nbr appno”
@ effdt
```

This creates two views:

- adm_appl_plan_s0 with all the records and adds “enddt” field
- adm_appl_plan_s1 with the current records (sysdate between effdt and enddt)

Higher level staging views

Add business rules and joins

```
-- academic work done at another institution
CREATE or REPLACE VIEW ext_acad AS
SELECT   d.*,
    unt_comp_total * decode(d.ext_term_type,'QTR',2/3,1) earned,
    decode(gpa_type,'HIGH',ext_gpa) hs_gpa
FROM ext_acad_sum_s1 s, ext_acad_data_s1 d
WHERE s.id(+)=d.id
   and s.ext_org_id(+)=d.ext_org_id
   and s.ext_career(+)=d.ext_career
   and s.ext_data_nbr(+)=d.ext_data_nbr;
```

Dual function example 1

Hide repetitive code -- void.sql and nv.sql

```
-- return Y if variable is null or empty or only contains spaces
CREATE or REPLACE FUNCTION void(val varchar) RETURN varchar2 as
BEGIN
      RETURN CASE WHEN nvl(length(trim(val)),0)=0 THEN 'Y' else 'N' end;
END;
/

-- return alternate value if variable is void (null or empty or only contains spaces)
CREATE or REPLACE FUNCTION nv(val varchar, altval varchar default ' ') RETURN varchar2 as
BEGIN
       RETURN CASE WHEN void(val)='Y' THEN altval ELSE val END;
END;
/
```

Dual function example 2

Encapsulating business rules

```
-- return a Banner-style term code for a given date
CREATE or REPLACE FUNCTION date2term (fdate date default null) RETURN varchar2 as

RESULT varchar2(6);
BEGIN
   SELECT to_char(nvl(fdate,sysdate),'rrrr')||
                CASE WHEN to_char(nvl(fdate,sysdate),'mm')<'05' THEN '20'
                WHEN to_char(nvl(fdate,sysdate),'mm')<'08' THEN '30'
                ELSE '40' END
        INTO result FROM dual;
   RETURN result;
end;
/
```

We have about 50 date, term, and year conversion functions:
https://sites.google.com/a/humboldt.edu/sdr/programming-information/date-and-term-functions

Staging function example

Encapsulating complex business rules

FUNCTION apstat(fid varchar,fterm varchar,fappno varchar) RETURN varchar;

This function returns an application status for a particular application.

1) Search through decision stack to status from most recent decision
2) If student has been admitted, see if this status needs to be modified
    a) Look for housing application
    b) Look for orientation registration
    c) Look for class enrollment

Building history tables (daily snapshots)

Using the application history table (app_sdr) as an example:

Begin by creating a table of the fields we want to capture.
This table contains 36 fields plus effdt, enddt, and obsolete

Create a view based on source data tables with the fields we are capturing: app_cs
This view contains data derived from 13 different CMS tables
and one foundation table

Call the change-data-capture script, cdc.sql, to add new or changed records to app_sdr
define tablename=app
@ cdc

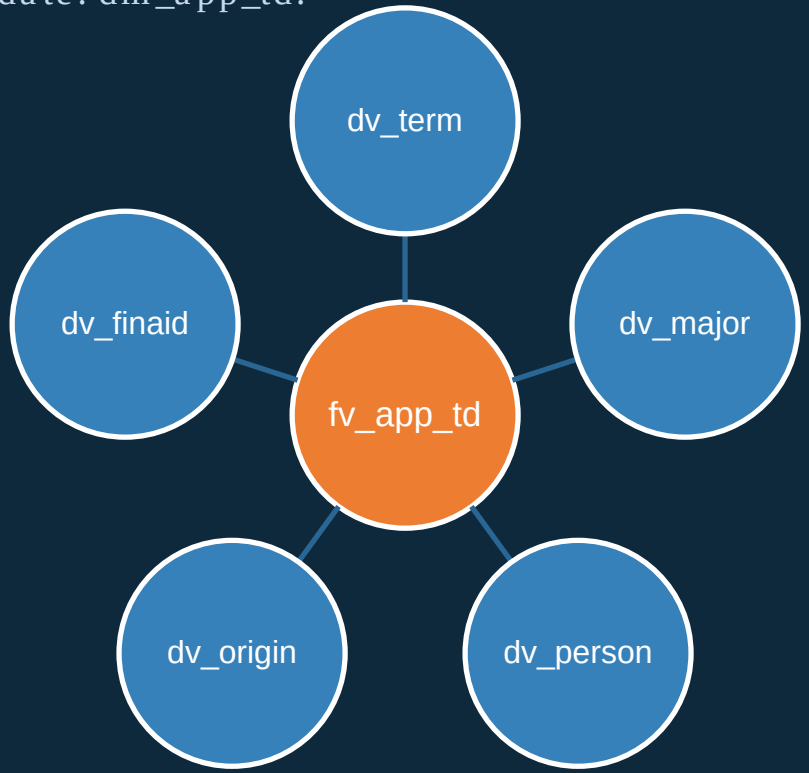
cdc.sql uses SQL set logic (MINUS) to compare the current data from the source tables with the last record with the same grain in history. Anything new or changed is added with today as the effective date. Data in history but missing from the source data is duplicated with today as effdt and obsolete=Y. The end date is then adjusted on all records.

History views -> Dimensional table

```
-- top-level history view
-- application facts to-date during the last 8 years

CREATE or REPLACE VIEW fact_app_td AS SELECT
    a.*
FROM fact_app_his a, term_td t
WHERE a.term_key=t.term_key
    AND t.comp_dt BETWEEN a.effdt AND a.enddt
    AND a.obsolete is null
/

-- “materialize” fact_app_td (as f_app_td) and build indexes on keys

DEFINE viewname=fact_app_td
@ dmgen
```

Combining dimensional views

Once all the dimension objects are created, they are combined into a dimensional model views. For example, applications to-date: dm_app_td.



CAIR
2016

Combining dimensional views (theoretical)

This is what we would like to write to create dm_app_td:

```
CREATE or REPLACE VIEW dm_app_td AS
SELECT   f.*,p.*,t.*,o.*,m.*,fa.*
FROM fv_app_td f, dv_person p, dv_term t, dv_origin o, dv_major m, dv_finaid fa
WHERE f.term_key=t.term_key
   AND f.person_key=p.person_key
   AND f.major_key=m.major_key
   AND f.origin_key=o.origin_key
   AND f.finaid_key=fa.finaid_key;
```

However, SQL would complain about the duplicate key fields, (and possibly others), so our actual view definitions are a bit more complicated.

Combining dimensional views (actual)

```
-- the first intermediate view, iv1_app_td.sql, joins the fact table to the first dimension

DEFINE viewname1=dv_term
DEFINE viewname2=fv_app_td
@ fieldsnotin

CREATE or REPLACE VIEW iv1_app_td AS SELECT  f.*, &f1&f2&f3&f4&f5
FROM fv_app_td f, dv_term d WHERE f.term=d.term_key;
@iv2_app_td
```

iv2_app_td adds dv_origin
iv3_app_td adds dv_person
Iv4_app_td adds dv_major
Iv5_app_td adds dv_finaid

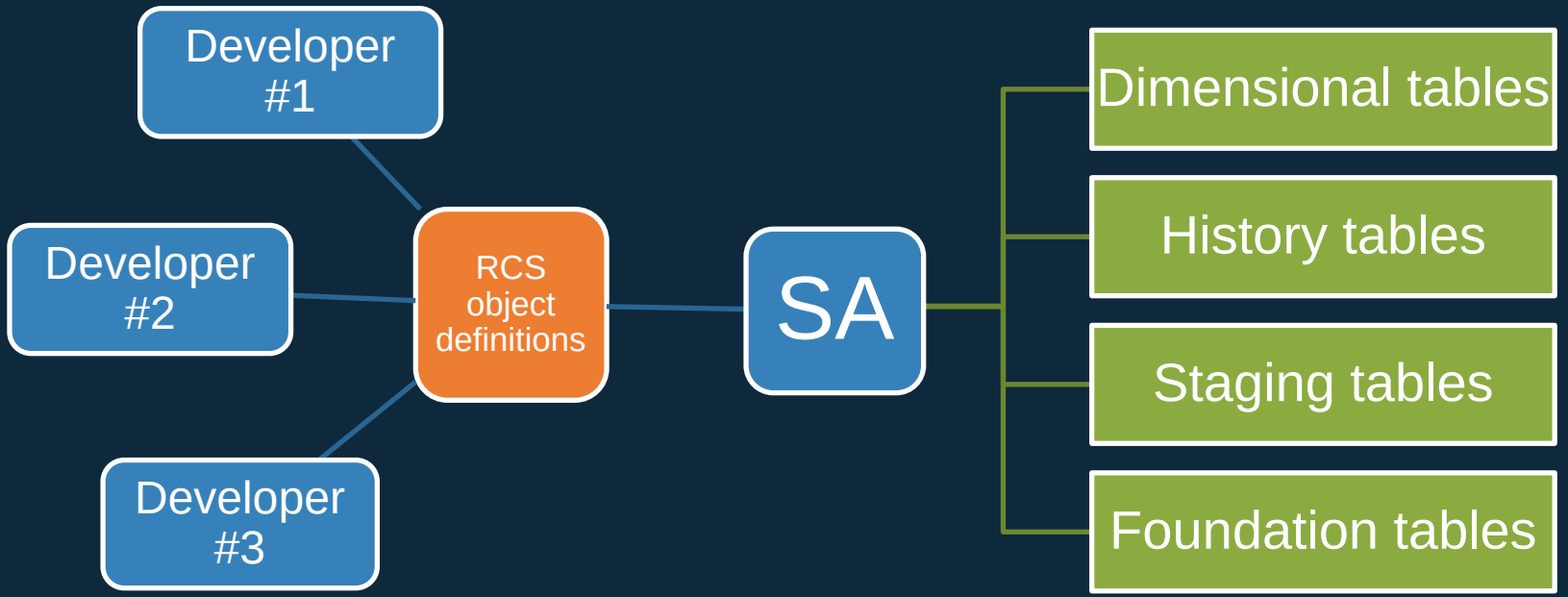
```
CREATE or REPLACE VIEW dm_app_td AS SELECT   f.* FROM iv6_app_td;
```

CAIR
2016

Development Environment

- Oracle Database (maintained by ITS)
  - SQL Developer, PL/SQL, SQL Plus
  - Production schema (SA) contains all tables and SQL objects
  - Developer schemas - copy of all SQL objects
- Linux front-end to Oracle DB Server (maintained by ITS)
  - RCS (revision control system) to store SQL object definition files
  - Emacs editor
  - Shell scripts that we write (example: ETL)
- 3rd-Party Tools
  - MobaXterm (SSH telnet) - free
  - Tableau - not free
  - MS Access - sort-of free (campus license)
  - Google site wiki - https://sites.google.com/a/humboldt.edu/sdr/

Development environment



CAIR 2016

Example of developing code

The California Graduation Initiative requires us to report URM. We already had URM in our data warehouse, but we had defined it differently than the state legislature.

- We considered anyone who was Hispanic, Black, Indian, or Pacific Islander to be URM, including those who were more than one race. We don't return a value for Unknown ethnicity or Nonresident Aliens.
- CA doesn't consider Pacific Islanders to be URM, nor does it believe anyone who is Two or More is URM, even if they are part Black or Indian. Plus, they include Nonresident Aliens and count Unknown as not URM.

To deal with this, I added a new field to the person dimension. First, I checked dv_person.sql out of RCS.

```
WARDHDEV> @ cout
File name: dv_person
v/RCS/dv_person.sql,v  -->  v/dv_person.sql
revision 1.18 (locked)
done
v/dv_person.sql has been checked out into /home/wwh7001/dev
```

Example of developing code (part 2)

I then edited dv_person.sql to add a new field: eth_urm_gi

```
...
CASE WHEN cit not in ('Y','I') or ethcode='8' then ' '
  WHEN instr(eth_indian||eth_black||eth_hispanic||eth_pacisl,'Y')>0 THEN 'Y'
  ELSE 'N' END eth_urm,
-- CA definition of URM for the Graduation Initiative
CASE WHEN ethcitcode in ('2','3','7') THEN 'Y'
  WHEN ethcitcode<'8' THEN 'N' END eth_urm_gi,
...
```

I then load dv_person.sql into my schema:

```
WARDHDEV> @ dv_person.sql
```

Since we have linked all our dependent views, this also loads all the views that depend on dv_person.

Example of developing code (part 3)

After checking for errors and testing the new field in dv_person and upstream dimensional models, I check dv_person back into RCS:

```
WARDHDEV> @ cin
```

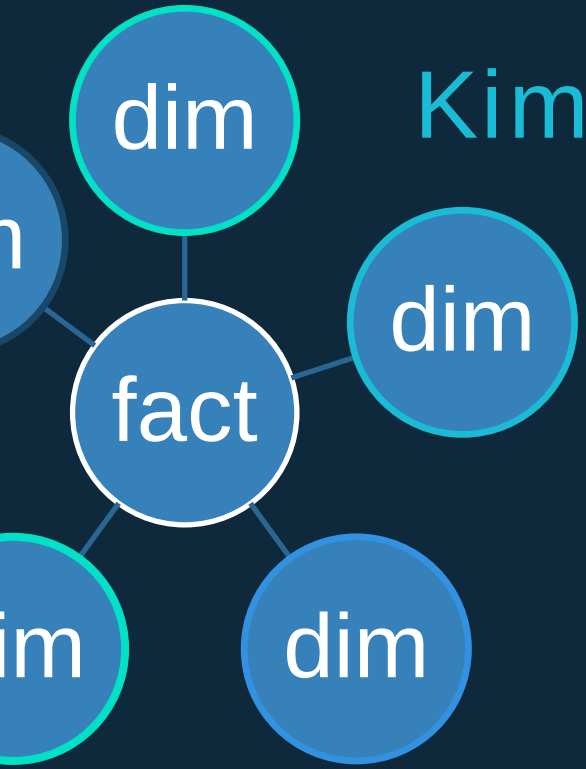
Then I refresh and revalidate all the SQL objects in my schema to make sure no errors have been introduced:

```
WARDHDEV> @ refresh
WARDHDEV> @ rev
pass 1 - revalidating 200 objects
```

Now I log into SA, the production account, and run refresh and revalidate again:

```
SA> @ refresh
SA> @ rev
pass 1 - revalidating 21 objects
```

The new field, eth_urm_gi, is now available to Tableau in all dimensional models that use dv_person.



Kimball Model Considerations

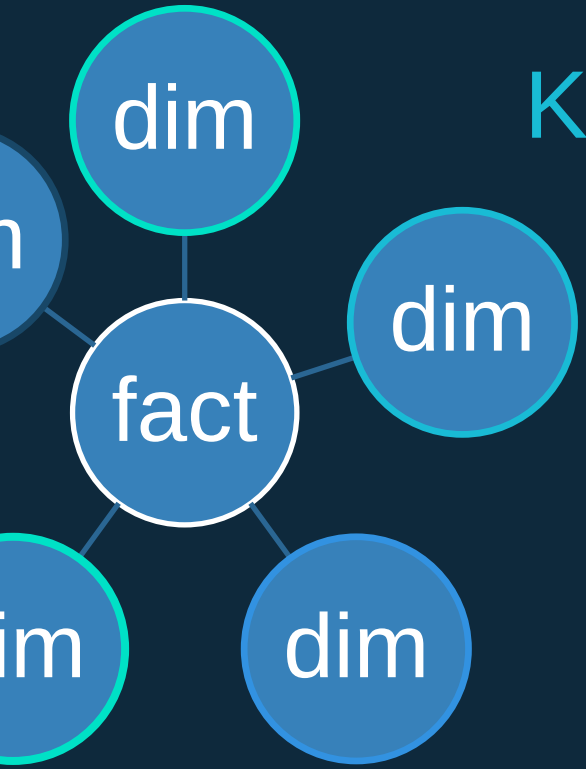
"Business Development Lifecycle Approach"

- Focus on adding *Business Value*
- *Dimensionally* structure the data that's delivered to the business
- Develop iteratively in manageable *life cycle* increments rather than attempting a galactic Big Bang approach

Goal: Delivering business intelligence to support University decision making.

CAIR 2016

http://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/dw-bi-lifecycle-method/



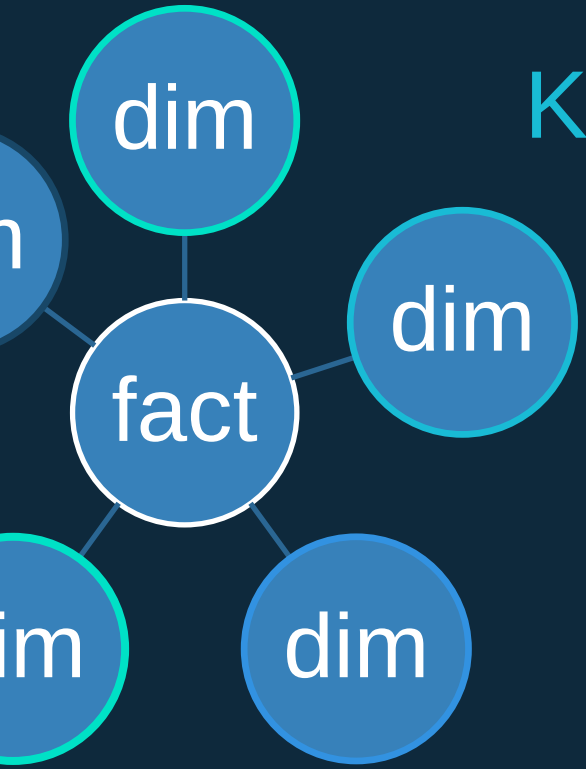
Kimball Model Deviations

Different approaches with a Common Ground:

- A data warehouse is a needed analytical environment for any organization.
- The goal is to publish the “right” data and make it easily accessible

CAIR 2016

http://www.kimballgroup.com/2004/03/differences-of-opinion/



Kimball Model Deviations

- Use of Views
- Surrogate Keys vs Native or Composite Keys
- Degenerate Dimensions - 1:1 Attributes stored with Fact

CAIR 2016

Conclusions

Pros of Building from “Scratch”

- Custom, Demand-based Design
- 3rd Party Independence
- Low start-up cost
- Broad knowledgebase (PL/SQL, C#, Java)

Pros of 3rd Party Software

- Visual Interface
- Version Control
- Metadata Support
- Debugging Support
- Transform Tools
- Multithreading

Thanks!

Any questions?

CAIR 2016

Credits

Presentation template by SlidesCarnival

Photographs by Unsplash