Introduction

School of Information Studies
Syracuse University

Agenda:

- The physical design process and environments
- Developing standards
- Physical design guidelines
- Demo: star schema generation from the detailed worksheet
- Demo: validating your star schema with data
- The test environment

So, Where Are We?

We covered:

- Dimensional modeling

We learned how to:

- Design dimensional models for relational databases

Detailed design

We'll cover

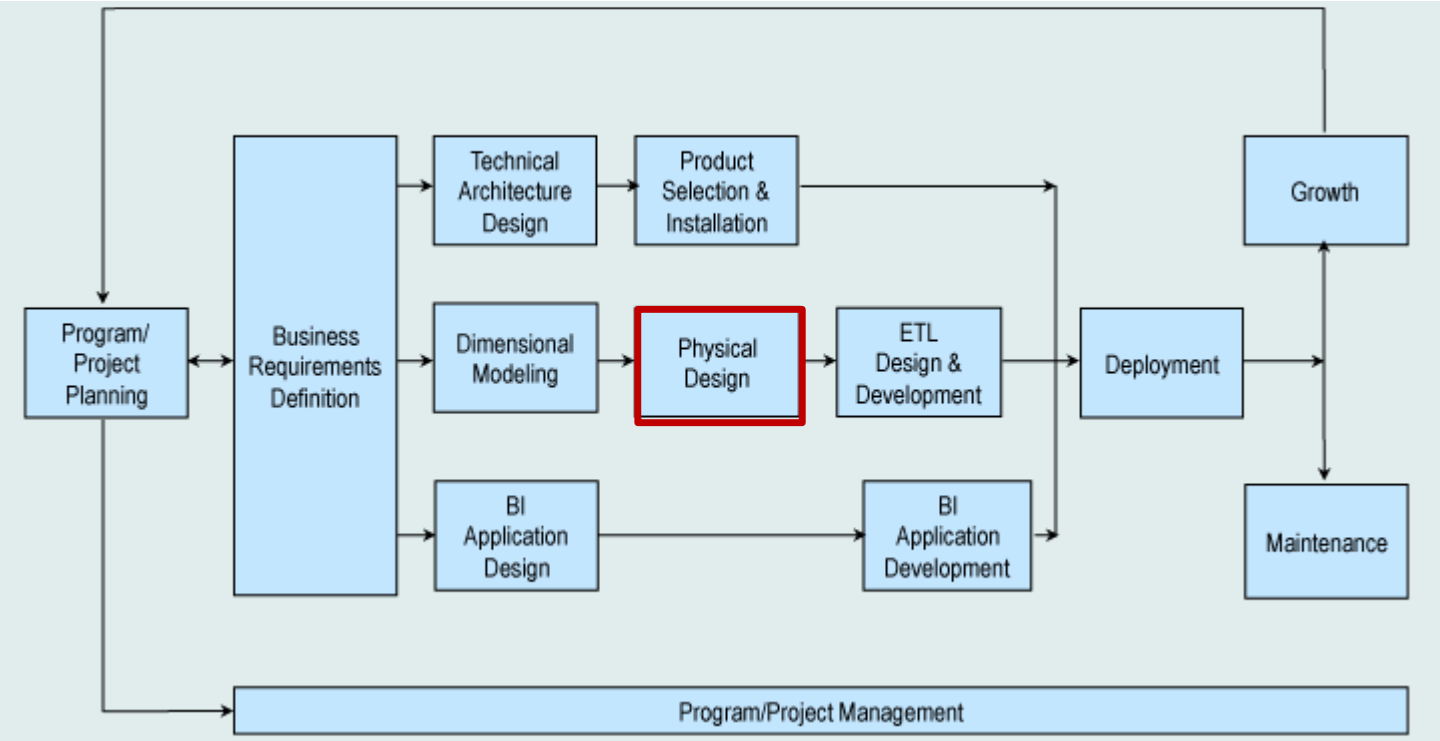
- ROLAP implementation of dimensional models

We'll learn how to

- Implement dimensional models in relational databases

Implementation

Kimball Lifecycle



The Goal: Detailed Design to ROLAP Implementation

Detailed Design

	A	B	J	K	L	M	N	O	P
1	Table Name	DimCustomer							
2	Table Type	Dimension							
3	Display Name	Customer							
4	Database Schema								
5	Table Description	Customers dimension							
6	Comment	comes from customers table in northwind							
7	Biz Filter Logic								
8	Size	one for each customer							
9	Generate Script?	N							
10									
11			Target						
12	Column Name	Display Name	Datatype	Size	Precision	Key?	FK To	NULL?	Default Value
13	CustomerKey	CustomerKey	int			PK ID		N	
	CustomerID	CustomerID	nvarchar	5				N	
14									
15	CompanyName	CompanyName	nvarchar	40				N	
16	ContactName	ContactName	nvarchar	30				N	
17	ContactTitle	ContactTitle	nvarchar	30				N	
18	CustomerCountry	CustomerCountry	nvarchar	15				N	
19	CustomerRegion	CustomerRegion	nvarchar	15				N	N/A'
20	CustomerCity	CustomerCity	nvarchar	15				N	
21	CustomerPostalCode	CustomerPostalCode	nvarchar	10				N	
	RowIsCurrent	Row Is Current	bit					N	TRUE
22									
	RowStartDate	Row Start Date	datetime					N	
23									
	RowEndDate	Row End Date	datetime					N	12/31/9999
24									
25	RowChangeReason	Row Change Reason	nvarchar	200				Y	
26	InsertAuditKey	InsertAuditKey	int			FK	DimAudit.AuditKey	N	
	UpdateAuditKey	UpdateAuditKey	int			FK	DimAudit.AuditKey	N	
27									

```
51 create table DimCustomer
52 (
53     CustomerKey int identity not null,
54     [Name] varchar(50) not null,
55     [City] varchar(20) not null,
56     [State] char(2) not null,
57     [ZipCode] char(5) not null,
58     constraint pkCustomerKey primary key (CustomerKey)
59 );
```

ROLAP

Today's Agenda:

Describe the process of implementing dimensional model designs in a relational database (ROLAP).

Discuss approaches to implementation.

Walk through an implementation together using a case study, so you can see this in action!

What Is Physical Design?

School of Information Studies
Syracuse University

What Exactly Is Physical Design?

- **Physical design** is the creation of actual artifacts as part of the design.
- In **data warehousing** the physical design is commonly implemented in a RDBMS.
- In the **Kimball enterprise bus technical architecture**, we implement **star schemas** with conformed dimensions.
- Physical design tasks include:
  - Making tables, keys, and constraints
  - Setting up schemas, synonyms, and views
  - Measuring and adjusting for performance
- Physical design is split up into the **internal (not user-facing)** and **external (user-facing)** models.

Internal vs. External Models

External Model

- User's view of the implementation
- Views, synonyms, natural/ business keys

Internal Model

- The actual implementation itself
- Tables, keys (PK/FK), triggers, check constraints

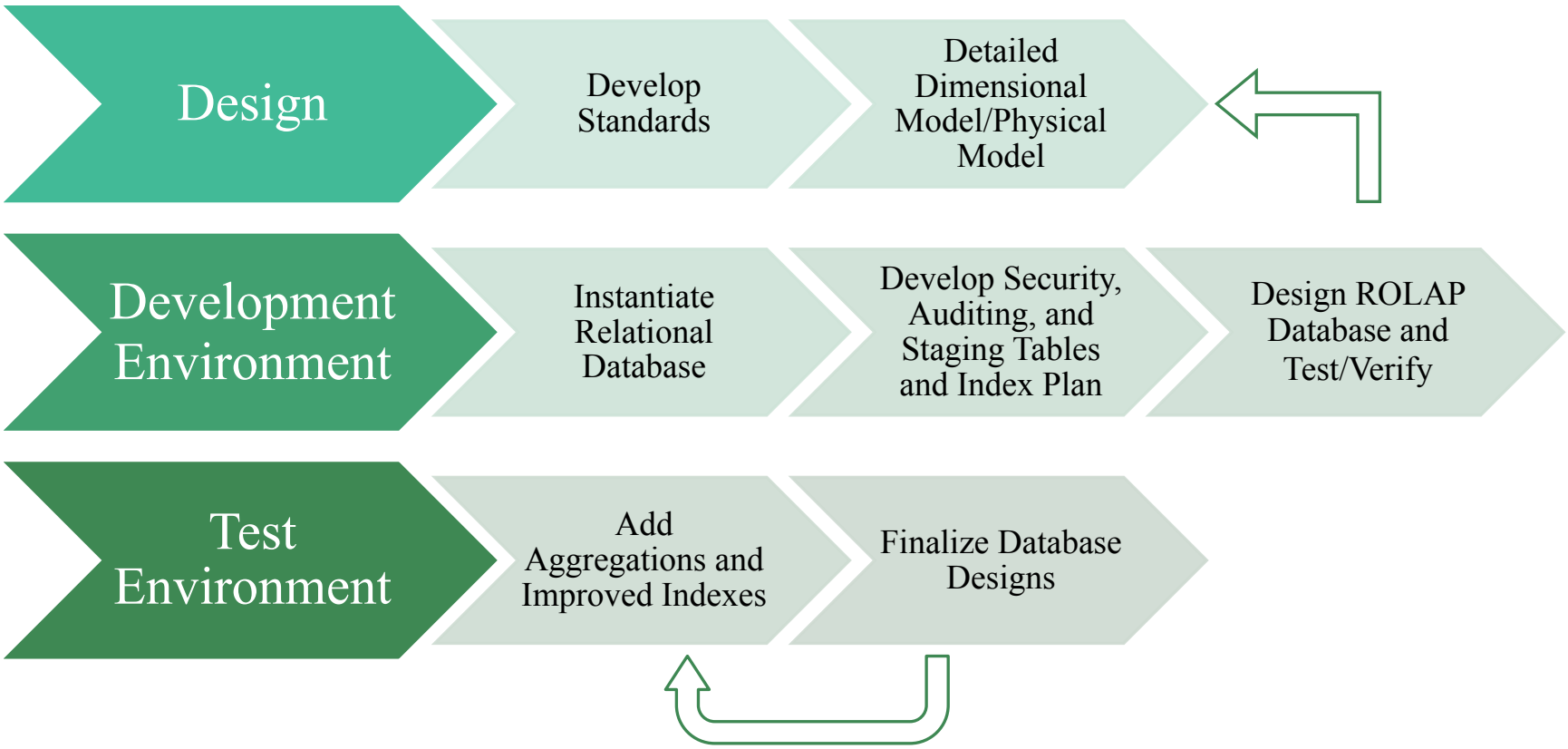
Conceptual to Physical: The Big Picture

	Conceptual Design	Logical Design	Physical Design
Task:	Identify functional requirements	Dimensional modeling	Create actual database objects
Tools used:	High-level dimensional modeling worksheet	Detailed-level dimensional modeling worksheet	SQL: database tables, keys, constraints, and views
Deliverable	Bus matrix	Dimensional models	ROLAP star schema

Development, Test, and Production Environments

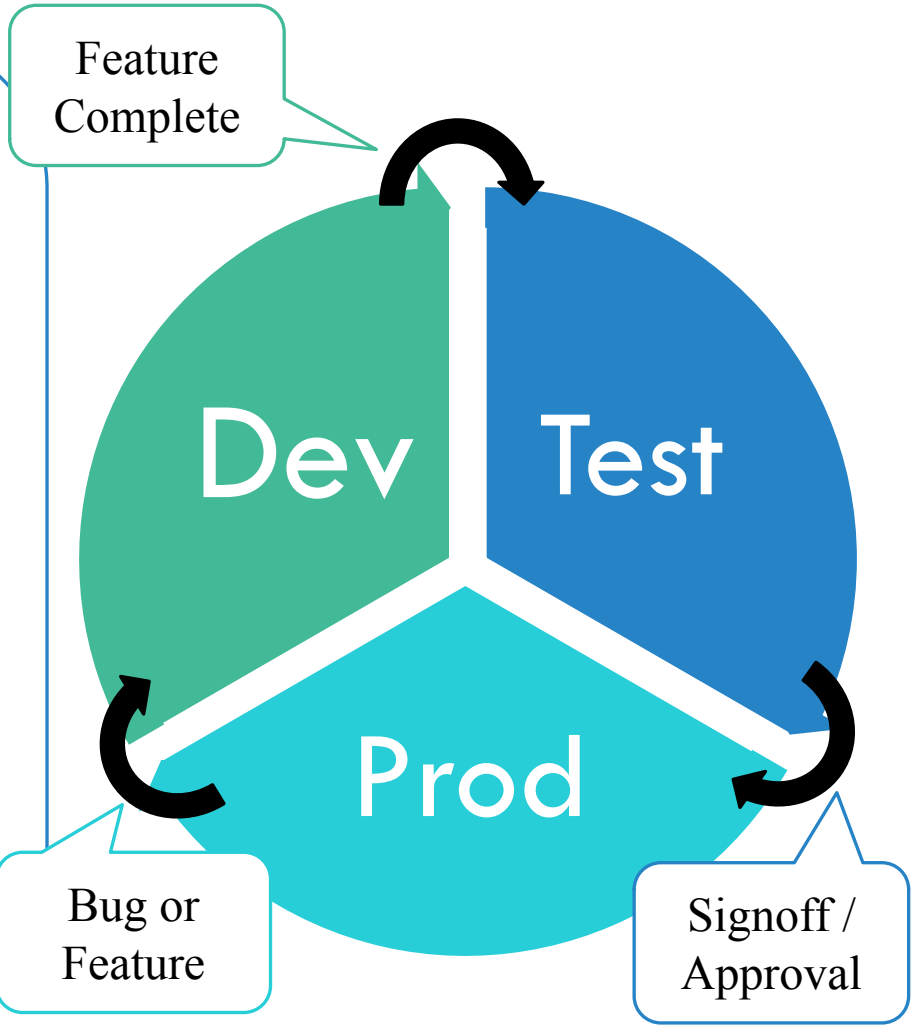
School of Information Studies
Syracuse University

The Physical Design Process at a Glance



A Word About Environments

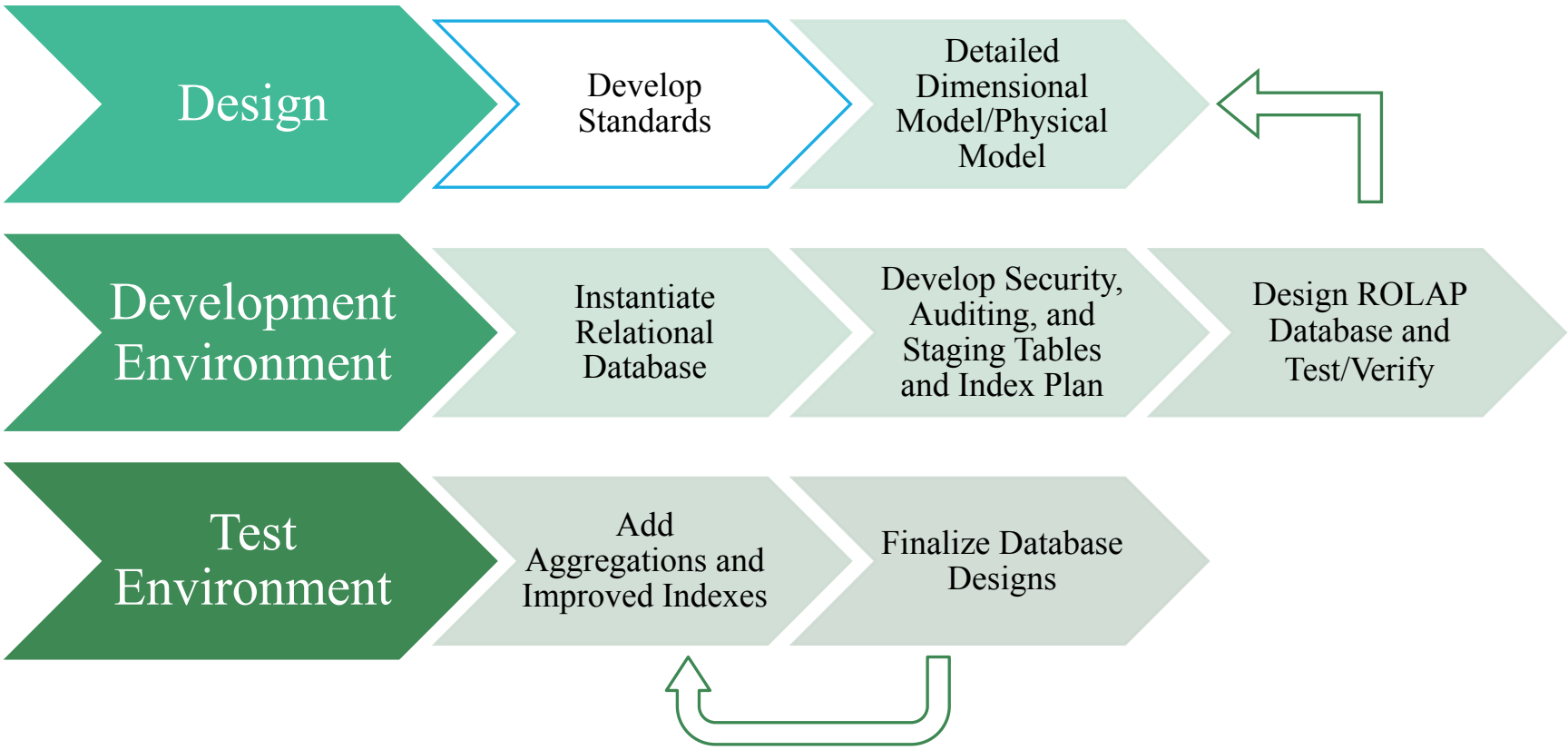
- **Development:**
  - Isolated to the developers only
  - Used for building things
  - Can use subsets of data
- **Test:**
  - Open to end users
  - As close to the production environment as possible
  - Measure performance here
- **Production:**
  - The actual system
  - Never make changes here



Developing Standards

School of Information Studies
Syracuse University

Developing Standards



Standards?

- Standards get everyone on the same page.
- Developers follow the standards, which leads to consistency.
- Business users are aware of the standards and require less training.
- Standards are challenging to enforce in large teams. It is best to put someone in charge of making sure the project teams adhere to standards.

Naming Conventions

School of Information Studies
Syracuse University

Naming Conventions

- Follow your organization's naming conventions.
- If you don't have any, establish them first!
- Consistency is key here!

How do you name dimensions?

- ✓ `Customer_Dim`
- ✓ `DimCustomer`
- ✓ `dim_customer`
- ✓ `[Dim Customer]`

Naming Conventions

- Table and view prefixes:
  - Stage table → **Stg**
    - Example: StgFudgemartEmployeeTimesheets
  - Dimension → **Dim**
    - Example: DimCustomer
  - Fact → **Fact**
    - Example: Fact EmployeeTimesheets
- Dimension keys:
  - Ends with "Key" → Customer**Key.**
  - Don't use ID; that's a convention for OLTP sources.

Is It Time to Use an SCM? Yes.

SCM ➔ Source code management

- Git, Subversion, Mercurial, CVS

Time to get serious about an SCM, since you'll be:

- Generating/creating code
- Making **lots of changes**
- Collaborating with others concurrently

SCM tools allow you to record and track changes to your code and easily rollback versions and collaborate with others.

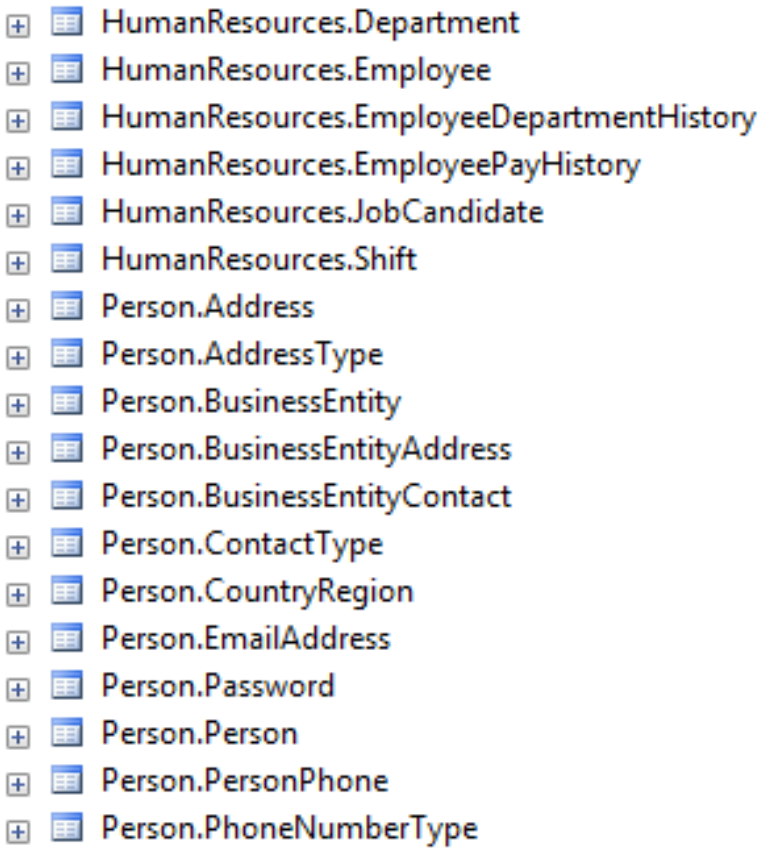
Schemas

School of Information Studies
Syracuse University

Schemas

- Schemas are **namespaces**, which facilitate the separation, management, and ownership of database objects.
- Objects are securable by schema.

```
CREATE SCHEMA name
```



Views and Synonyms

School of Information Studies
Syracuse University

Synonyms and Views

- Synonyms and views are logical abstractions of tables and SQL SELECT statements, respectively.
- For any table **directly accessible by an end user**, a view or synonym should be used. This helps us build the **external model**.
- This way you can change the underlying tables without affecting the **external dependencies** (reports, BI applications, etc.).

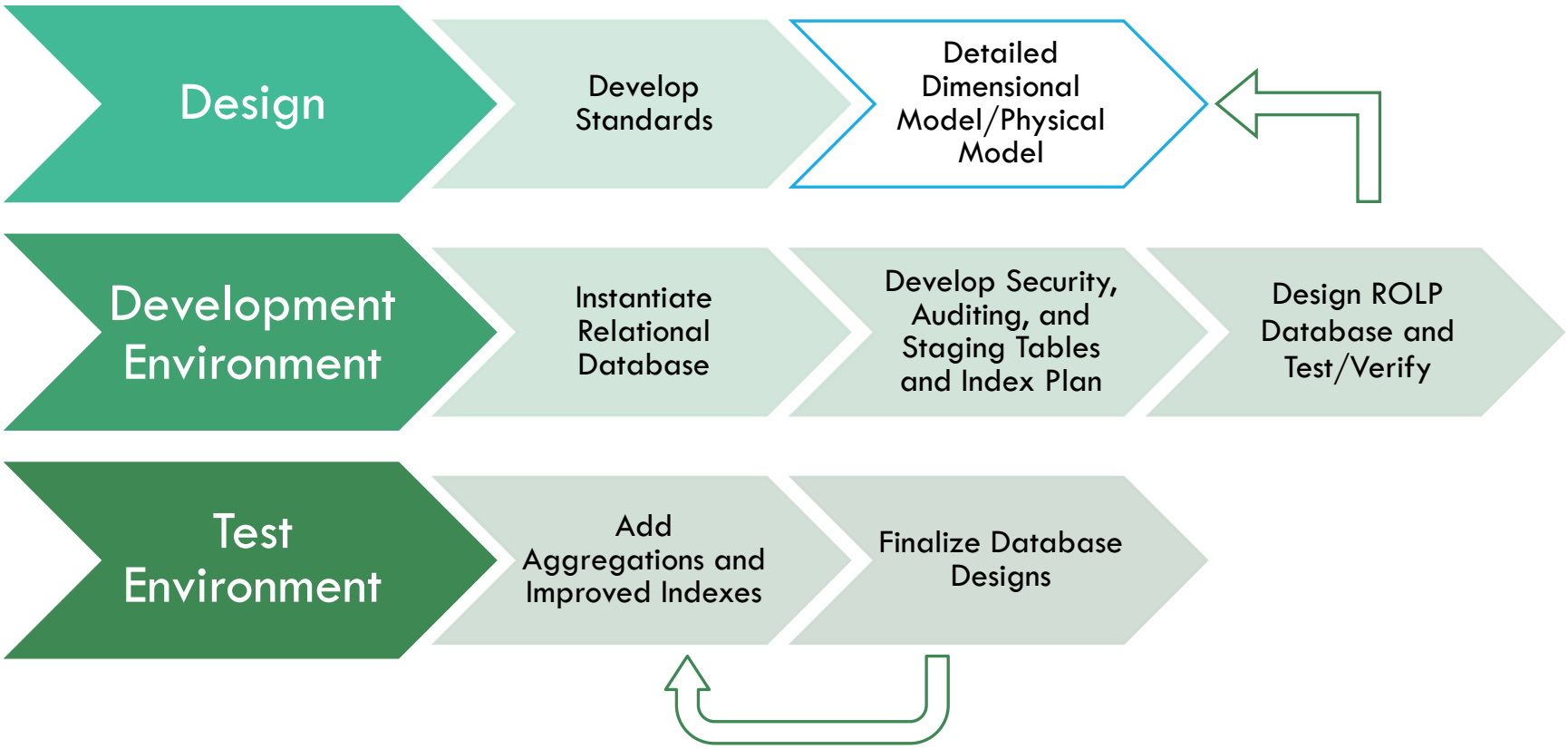
```
CREATE VIEW name AS …

CREATE SYNONYM name FOR …
```

Data Type Selection

School of Information Studies
Syracuse University

The Physical Model



Every Byte Counts in a Fact Table!!!

	Total bytes of data types	88 Bytes
X	Total number of rows	5 Billion
	Total amount of disk storage for this fact table	**440 GB**

Data Types

Data Type	Description	Storage
`smallint`	Integers -32K to +32K	2 bytes
`int`	Integers -2B to +2B	4 bytes
`bigint`	Integers $-9B^2$ to $+9B^2$	8 bytes
`decimal(p,s)`	Decimal values from -10^{38} to $+10^{38}$	5-17 bytes
`money`	Monetary values -900T to + 900T	8 bytes
`datetime`	Date/times from 1/1/1753 to 12/31/9999	8 bytes
`char(n)`	Fixed-character string specific charset	n bytes
`nchar(n)`	Fixed-character string Unicode charset	2n bytes
`varchar(n)`	Variable length string specific charset	2 + n bytes
`nvarchar(n)`	Variable length string Unicode charset	2 + 2n bytes

Saving Space With Types

- Don't use money or decimal when int will suffice.
  - Savings: 1 to 13 bytes
- Don't use nvarchar unless you require Unicode.
  - Savings: 4 bytes
- Don’t use char(1) for Yes/No; use bit.
  - Savings: 0 to 7 bytes (depending on the number of bit fields)
- Use smallint instead of int.
  - Savings: 4 bytes
- Use int YYYYMMDD for date instead of datetime.
  - Savings: 4 bytes

Primary and Foreign Keys

School of Information Studies
Syracuse University

Primary Keys

- Dimension tables should use **surrogate keys.**
- Fact tables should use **composite keys** made up of **dimension foreign keys** and **degenerate dimensions**.
- Surrogate keys can be used in the fact table, but they increase the table size and do not improve performance.
- Surrogate keys are **number sequences. Date surrogate** keys should be of the form YYYYMMDD.
- ***Every byte counts in your fact table!***

Foreign Keys

- **Foreign keys** are important. Don't devalue!
- FKs enforce **referential integrity** between the PK in the dimension table and the FK in the fact table.
- ***This prevents you from inserting invalid data into the fact table.***
- If you're concerned about the **performance impacts of constraint checking,** you can drop the FKs, insert the data, then reinstate the constraints with the `no-check` option.

Example: Primary and Foreign Keys

	CustomerKey	CustomerID	CompanyName	ContactName	ContactTitle	CustomerCountry	CustomerRegion	Custome
1	65	QUICK	QUICK-Stop	Horst Kloss	Accounting Manager	Germany	N/A	Cunewa
2	67	RATTC	Rattlesnake Canyon Grocery	Paula Wilson	Assistant Sales Representative	USA	NM	Albuque
3	50	LONEP	Lonesome Pine Restaurant	Fran Wilson	Sales Manager	USA	OR	Portland
4	88	WANDK	Die Wandernde Kuh	Rita Müller	Sales Representative	Germany	N/A	Stuttgar
5	60	PERIC	Pericles Comidas clásicas	Guillermo Fernández	Sales Representative	Mexico	N/A	México
6	16	CHOPS	Chop-suey Chinese	Yang Wang	Owner	Switzerland	N/A	Bern
7	64	QUEEN	Queen Cozinha	Lúcia Carvalho	Marketing Assistant	Brazil	SP	Sao Pau
8	43	LAMAI	La maison d'Asie	Annette Roulet	Sales Manager	France	N/A	Toulous
9	62	PRINI	Princesa Isabel Vinhos	Isabel de Castro	Sales Representative	Portugal	N/A	Lisboa
10	46	LEHMS	Lehmanns Marktstand	Renate Messner	Sales Representative	Germany	N/A	Frankfu

	ProductKey	OrderID	CustomerKey	EmployeeKey	OrderDateKey	ShippedDateKey	Quantity	ExtendedPriceAmount	DiscountAmount	SoldAmount
1	1	10285	65	1	19960820	19960826	45	648.0000	129.6000	518.4000
2	1	10294	67	3	19960830	19960905	18	259.2000	0.0000	259.2000
3	1	10317	50	5	19960930	19961010	20	288.0000	0.0000	288.0000
4	1	10348	88	3	19961107	19961115	15	216.0000	32.4000	183.6000
5	1	10354	60	7	19961114	19961120	12	172.8000	0.0000	172.8000
6	1	10370	16	5	19961203	19961227	15	216.0000	32.4000	183.6000
7	1	10406	64	6	19970107	19970113	10	144.0000	0.0000	144.0000
8	1	10413	43	2	19970114	19970116	24	345.6000	0.0000	345.6000
9	1	10477	62	4	19970317	19970325	15	216.0000	0.0000	216.0000
10	1	10522	46	3	19970430	19970506	40	720.0000	144.0000	576.0000

Nulls and Unknown Members

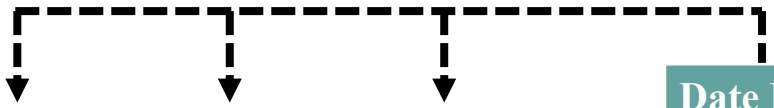
School of Information Studies
Syracuse University

Best Practices for Nulls

- The **attributes** in your **dimension** tables should not have nulls.
- Attributes **without a value** (null) should be assigned a value. Why is it null?
  - Example: no e-mail address? ➔ "No E-mail"
- **Foreign keys** in the **fact** table should never be null; they should be assigned an **unknown member.**
- Nulls are okay for ***values*** in the fact tables. Null valued facts are not included in calculations.

Unknown Members

- We should not insert NULL into the FK of a fact table. We should use an unknown member instead!
- These are special coded values, which indicate why there is no value.
- Every dimension should have at least one unknown member.



Order ID	Order Date Key	Ship Date Key	Delivery Date Key
99001	20170401	20170403	20170412
99002	20170408	20170409	-3
99003	20170412	-2	-3
99004	20170415	-2	-3
9905	-1	-1	-1

Date Key	Date	Date Name
-3	0000-00-00	Not Delivered
-2	0000-00-00	Not Shipped
-1	0000-00-00	Unknown Date
20100101	2010-01-01	1/1/2010
20100102	2010-01-02	1/2/2010

Implementing Type 2 SCD Metadata

School of Information Studies
Syracuse University

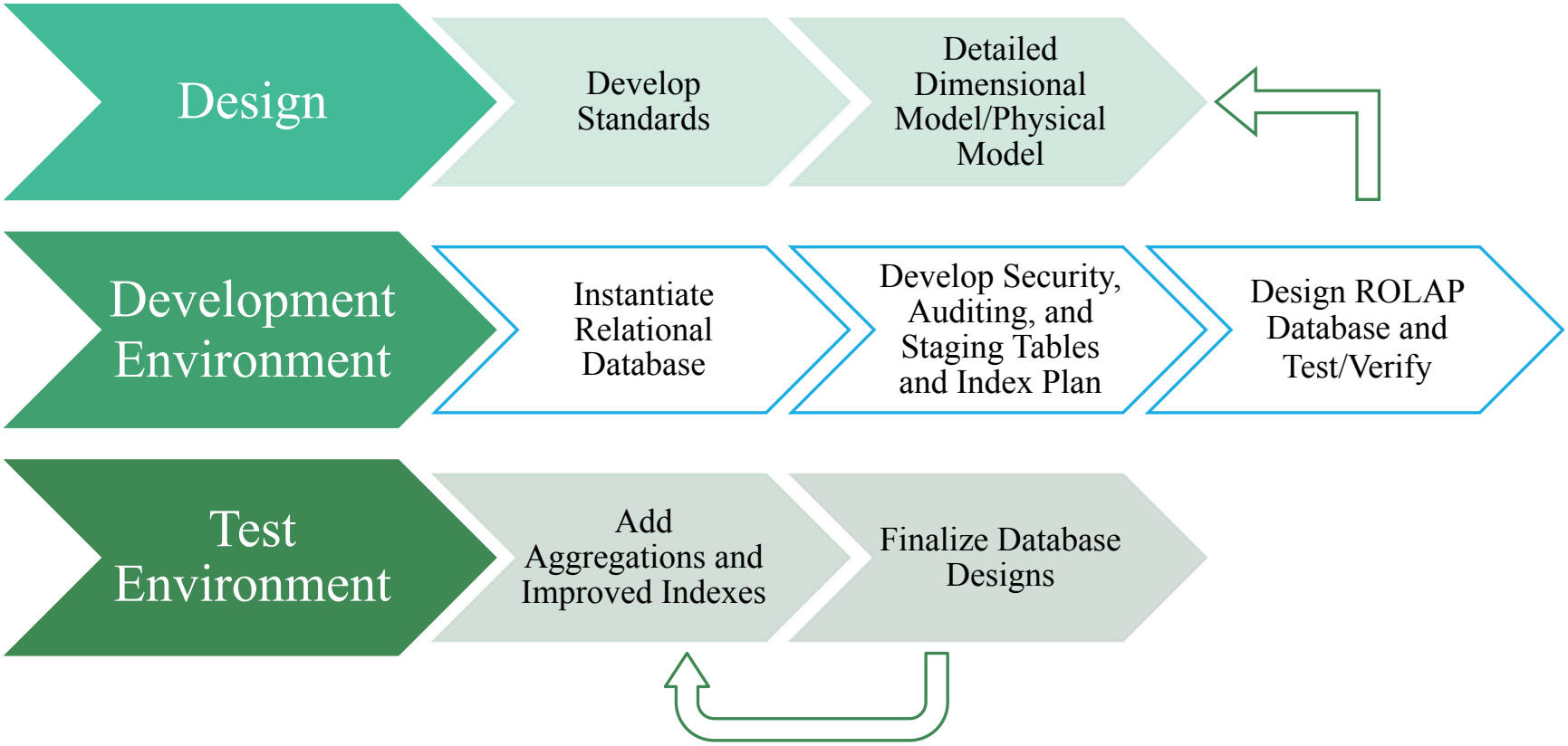
Type 2 SCD Metadata

- Type 2 preserves history.
- These columns should be added to assist with tracking but not displayed to the end user.
- Perfect use case for a view to separate the table from the user-facing columns.
- Add these columns:
  - RowIsCurrent (Y/N) → Is this the current row?
  - RowStartDate (datetime) → Start date of valid row
  - RowEndDate (datetime) → End date of valid row
  - RowChangeReason (text) → Explain why row changed

The Development Environment

School of Information Studies
Syracuse University

Build Your Development Environment



Instantiate the ROLAP Database

- You'll need this **before** you can develop the ETL process.
- You don't need to focus on **performance** at this point because you don't know the **bottlenecks**.
- The Development environment should be separate from the test environment.
- Update your detailed modeling worksheet as you make changed to keep your schema consistent with the documentation.

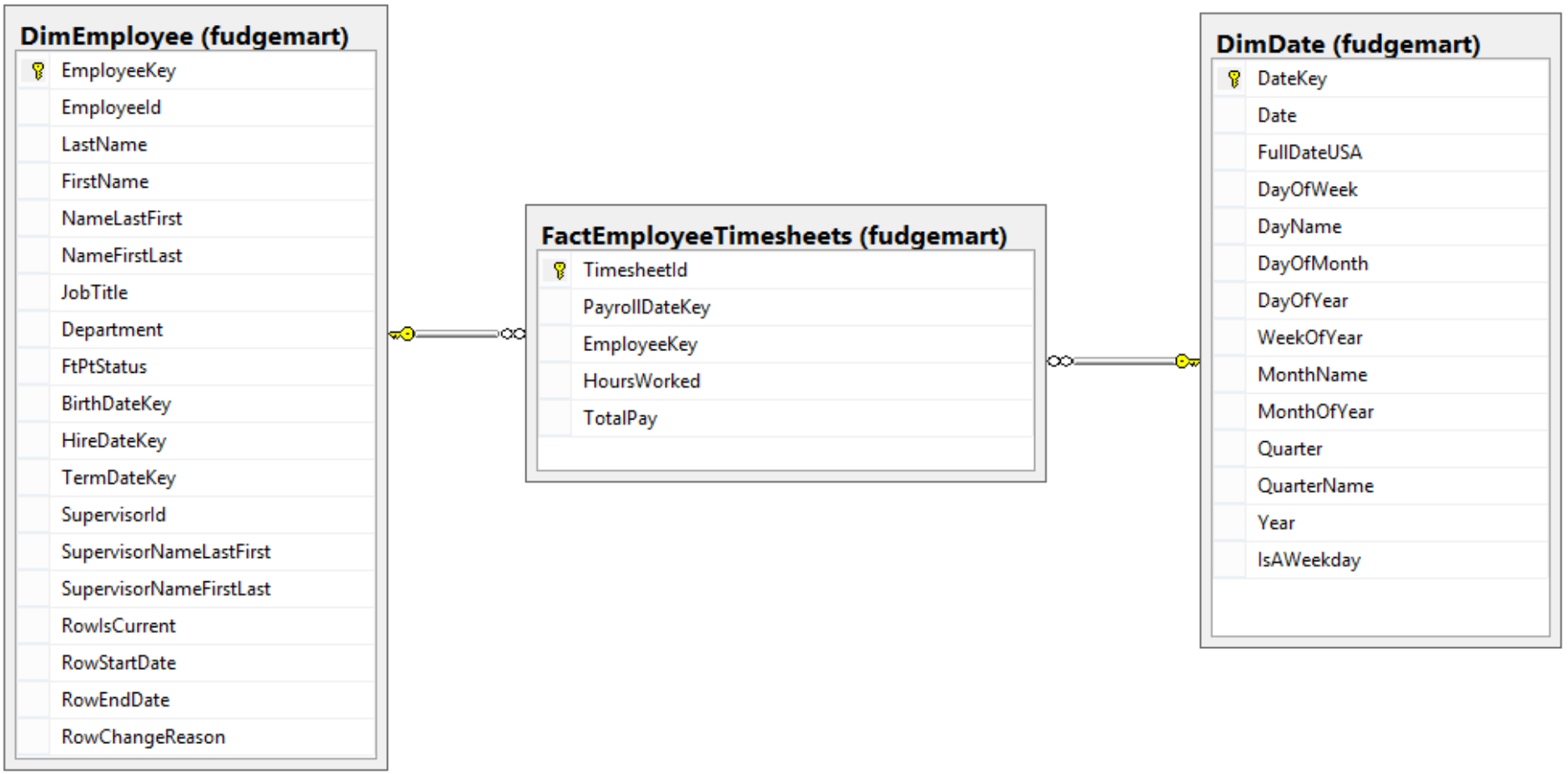
Our Case Study:
Fudgemart Employee Time Sheets

Business Process Name	Fact Table	Fact Grain Type	Granularity	Facts	Payroll Date	Employee
Employee Timesheets	FactEmployeeTimesheets	Transation	One row per timesheet entry (employee + date)	Hour Worked, Total Pay	X	X

We will:

- ✓Implement the ROLAP Schema
- ✓Load with data to test / verify the model

The ROLAP Star Schema



Overview of the Process

School of Information Studies
Syracuse University

Initial ETL Load for Model Validation

Doing initial with SQL: Why?

1. Stage data.
   - Address data-sizing issues.
   - Take samples/subsets.
2. Transform and load into model.
   - Understand the required manipulations from source to target.
   - You'll be better prepared for the actual ETL.
3. Test and verify the model.
   - Fact grain makes sense?
   - Facts are additive?
   - Default members in place of null?
   - Dimensions attributes have values?

Best Practices for Data Extraction to Stage

- Always stage your data "as is" to avoid a dependency on the source systems.
- This is important during development as live data are always changing, and we want to freeze data to a point in time during initial development.
- You do not want your stage data in the same database or schema as your data warehouse. Helps keep the models "tidy." **You should be working in a development environment.**

Dataset Too Big for Your Development Environment?

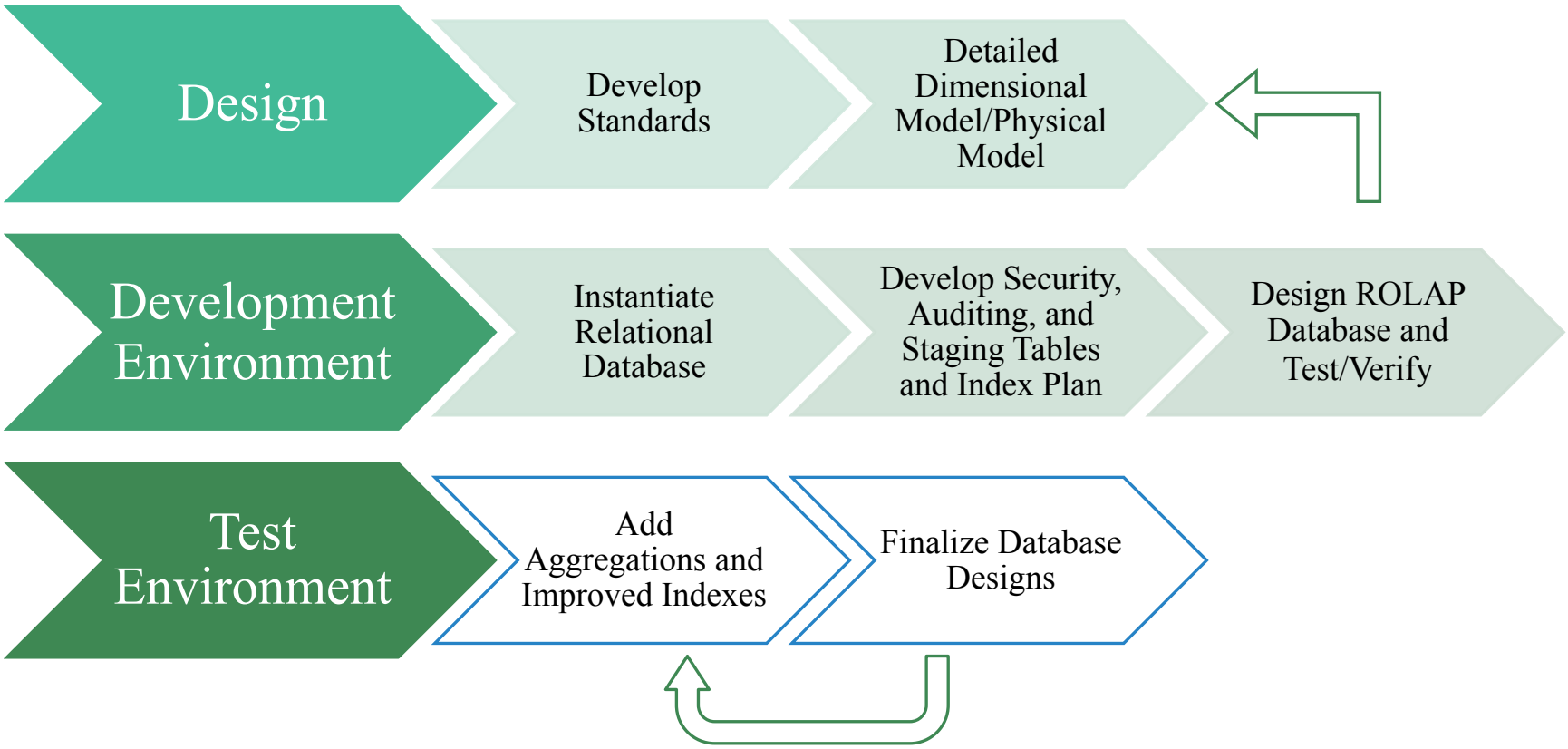
Create a subset!

- Random sample of transactions.
  - Get 10% of the 1,000,000 rows.
- Filter on a specific time.
  - One year rather than all 10 years.
- Random sample from a dimension.
  - Select 10 random customers.
- Filter on a dimension.
  - Only specific customers or products
- **When you move to test, you must use ALL the data!**

The Test Environment

School of Information Studies
Syracuse University

The Test Environment



Test Environment

- Must be a networked environment so users can access it.
- End users and key stakeholders are doing the testing.
- Same dataset as production environment (or as close to it as feasible).
- Typically just an additional database instance on the production server but can be a separate installation.
- Is loaded with only the data needed for testing. When testing is complete, the environment is reset.
- Key tasks: **monitor usage** and **adjust system performance.**

Test Environment Credo

"Your test environment contains ONLY what you're testing, but ALL of what you're testing."

Example:

- We are testing Fudgemart employee timesheets, so ONLY need that star schema, but we need ALL the data loaded into it.
- It's about performance!

Performance

Ways to Measure

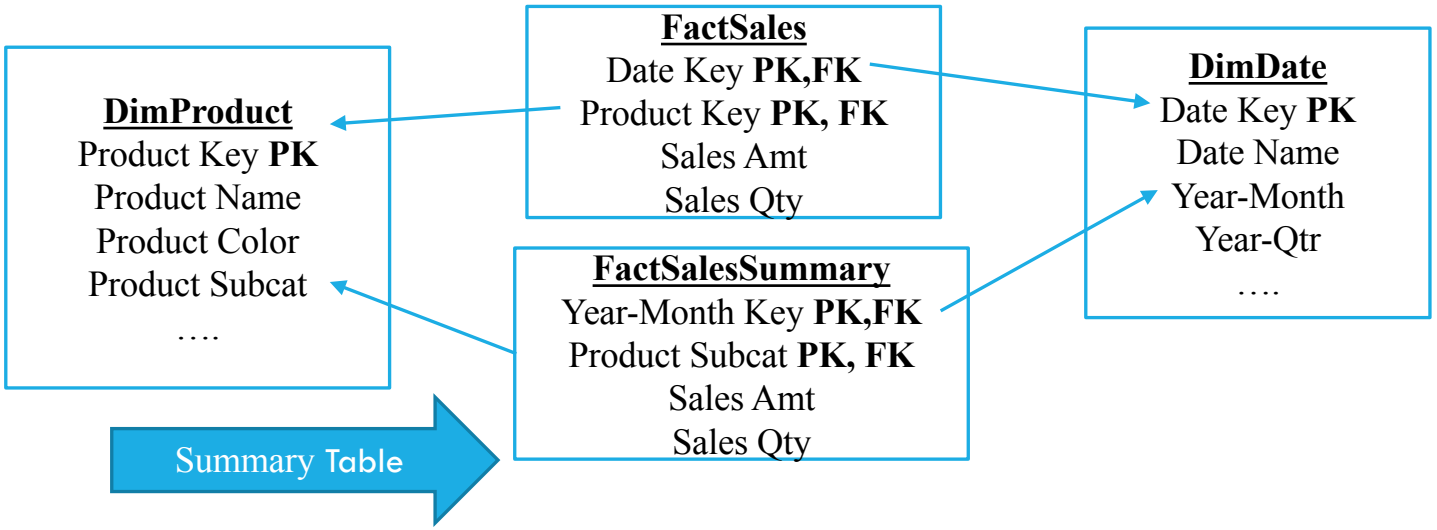
- Query execution plans
  - What has run?
- Query execution times
  - How long does it take?

Ways to Improve

1. Summary tables
2. Partitioning
3. Indexes

Build Summary Tables

- Number-one way to improve performance.
- Aggregate popular roll-up data.
- Monitor queries to find out what's popular.
- Created through the ETL process.



Materialized Views

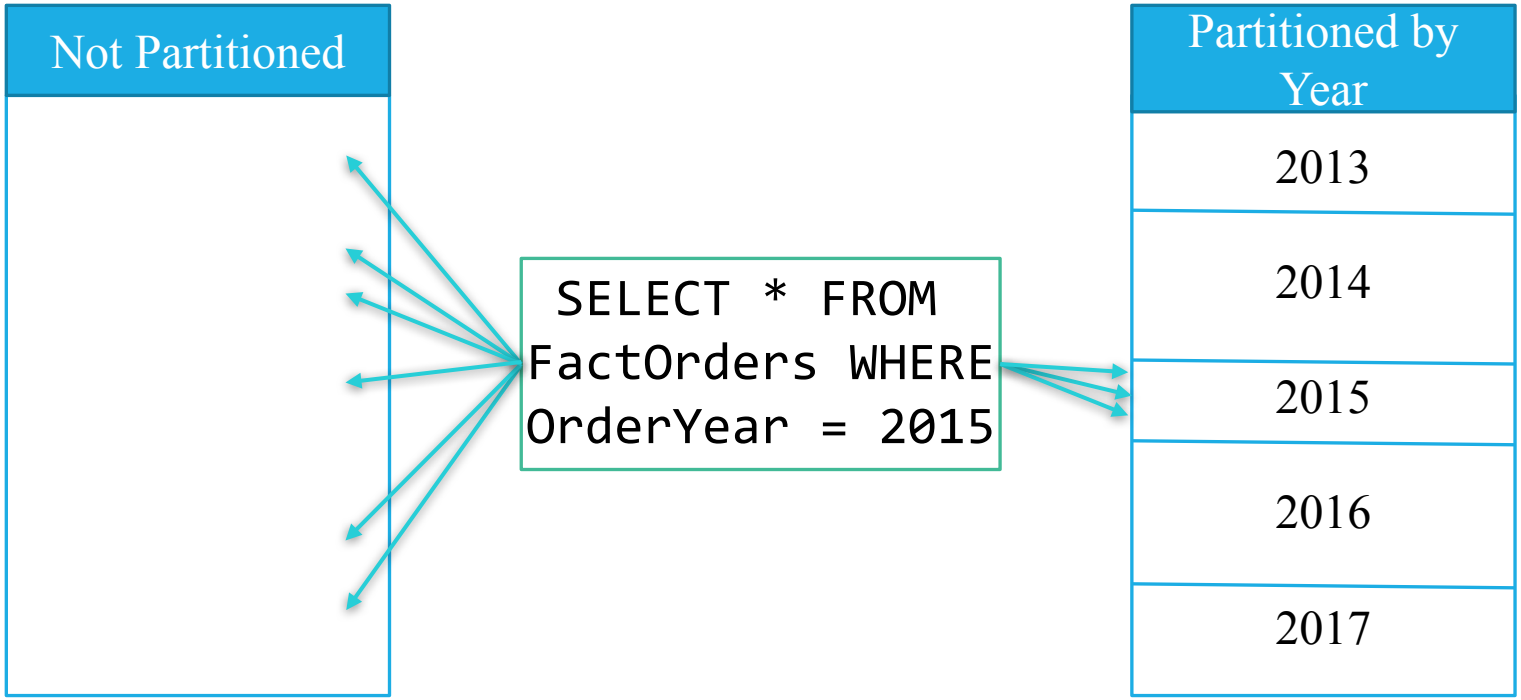
- An SQL view that saves a copy of the query output.
- This offers a performance boost because the materialized view output acts as a table.
- SQL server offers indexed views, which function similarly.
- Alternative to summary tables, which must be created each time.

Partitioning

School of Information Studies
Syracuse University

Partitioning

- Number-two way to improve performance.
- Most data in fact tables are inserted in chronological order.
- It makes sense to physically organize the fact table by date.



Indexing and Clustered Index

School of Information Studies
Syracuse University

Index

- Number-three way to improve performance.
- **Index** improves search of a table by creating an internal structure with keys built from one or more columns. You can have multiple indexes per table.
- **Clustered index** is an index on the table itself. It determines the order the data is written to the table; thus there can only be one.

DimProduct		
ProdKey	Name	Department
1	Hammer	Hardware
2	T-Shirt	Clothing
3	Wrench	Hardware
4	Socks	Clothing
5	TV Set	Electronics
6	Shoes	Clothing
7	Drill	Hardware

Department Index	
Department	IDs
Clothing	2,4,6
Electronics	5
Hardware	1,3,7

As data change, the index must be rebuilt, hence the negative impact on write performance.

Indexing Dimension and Fact Tables

- If your DBMS supports **bitmapped indexes**, add them to your dimension tables on attributes involved in **row filters**.
- For fact tables, follow the index plan optimizer of your DBMS.