

Relational Data Model: Part 2

“Bad” Relational Schema

- ❖ Suppliers(name, address, item, price)
- ❖ Issues
 - ❖ redundancy - address is repeated for every item
 - ❖ wasted space - is that still a concern?
https://databasemanagement.fandom.com/wiki/Category:Data_Anomalies
 - ❖ wasted resources (time, compute) to maintain it
 - ❖ potential inconsistency (update anomalies) - may update address in one place but not another
 - ❖ insertion anomalies - can't have a supplier without having at least one item
 - ❖ NULLs an option but create their own problems
 - ❖ what if a real item gets added, do we delete NULL valued row
 - ❖ (Item, Name) is a key and having NULL values in key is “not-good”
 - ❖ deletion anomalies - reverse (if all items deleted we lose address info)

Anomalies

- ❖ update anomalies - may update address in one place but not another
- ❖ insertion anomalies - can't have a supplier without having at least one item
 - ❖ NULLs an option but create their own problems
 - ❖ what if a real item gets added, do we delete NULL valued row
 - ❖ (Item, Name) is a key and having NULL values in key is "not-good"
- ❖ deletion anomalies - reverse (if all items deleted we lose address info)

name	addr	item	price
S1	123 Any	P1	10
S1	123 Any	P2	20
S2	1 Main	P1	11
S2	1 Main	P3	100
S4	11 State	P1	9
S3	10 State	P4	1000

Update Anomaly

- ❖ update s set $addr =$
'100 Any' where $item =$
'P1' and $name =$ 'S1'

name	addr	item	price
S1	123 Any	P1	10
S1	123 Any	P2	20
S2	1 Main	P1	11
S2	1 Main	P3	100
S4	11 State	P1	9
S3	10 State	P4	1000

name	addr	item	price
S1	100 Any	P1	10
S1	123 Any	P2	20
S2	1 Main	P1	11
S2	1 Main	P3	100
S4	11 State	P1	9
S3	10 State	P4	1000

Insertion Anomaly

- ❖ insertion anomalies - can't have a supplier without having at least one item
 - ❖ NULLs an option but create their own problems
 - ❖ what if a real item gets added, do we delete NULL valued row
 - ❖ (Item, Name) is a key and having NULL values in key is "not-good"
- ❖ want to add 'S5'
- ❖ add a "real" entry for S5
- ❖ add a part with no supplier

name	addr	item	price
S1	123 Any	P1	10
S1	123 Any	P2	20
S2	1 Main	P1	11
S2	1 Main	P3	100
S4	11 State	P1	9
S3	10 State	P4	1000

name	addr	item	price
S1	100 Any	P1	10
S1	123 Any	P2	20
S2	1 Main	P1	11
S2	1 Main	P3	100
S4	11 State	P1	9
S3	10 State	P4	1000
S5	100 Main	null	null
S5	100 Main	P2	22
null	null	P5	null

Deletion Anomaly

- ❖ deletion anomalies - can't have a supplier without having at least one item
 - ❖ NULLs an option but create their own problems
 - ❖ what if a real item gets deleted, do we delete NULL valued row
 - ❖ (Item, Name) is a key and having NULL values in key is "not-good"
- ❖ delete 'P4' but lose 'S3'
- ❖ could make it null
- ❖ same issue deleting say 'S2'

name	addr	item	price
S1	123 Any	P1	10
S1	123 Any	P2	20
S2	1 Main	P1	11
S2	1 Main	P3	100
S4	11 State	P1	9
S3	10 State	P4	1000

name	addr	item	price
S1	100 Any	P1	10
S1	123 Any	P2	20
S2	1 Main	P1	11
S2	1 Main	P3	100
S4	11 State	P1	9
S3	10 State	P4	1000

Better Way

- ❖ Break it up as:
 - ❖ SuppAddr(name, address)
 - ❖ SuppItemPrice(name, item, price)
- ❖ Now we'll need a join to discover address for an item
- ❖ But problems go away

name	addr
S1	123 Any
S3	10 State
S2	1 Main
S4	11 State

name	item	price
S2	P3	100
S1	P2	20
S1	P1	10
S3	P4	1000
S2	P1	11
S4	P1	9

Update Anomaly?

- ❖ update s set $addr =$
'100 Any' where $name$
 $= 'S1'$

name	addr
S1	100 Any
S3	10 State
S2	1 Main
S4	11 State

name	item	price
S2	P3	100
S1	P2	20
S1	P1	10
S3	P4	1000
S2	P1	11
S4	P1	9

Insertion Anomaly?

- ❖ want to add 'S5'
- ❖ add a “real” entry for S5
- ❖ add a part with no supplier
 - ❖ use null
 - ❖ have separate table, if part can be an independent entity

name	addr
S1	100 Any
S3	10 State
S2	1 Main
S4	11 State
S5	100 Main

name	item	price
S2	P3	100
S1	P2	20
S1	P1	10
S3	P4	1000
S2	P1	11
S4	P1	9
S5	P2	22
null	P5	null

Dependency

- ❖ Functional dependency says that if we know value of a set of columns uniquely determines the value of another set of columns.
 - ❖ Thus it's a statement (constraint) saying what relation instances are allowed
- ❖ Cause and cure of redundancy go hand-in-hand: we use the functional dependency to eliminate / reduce redundancy

Constraints Revisted

- ❖ Domain constraints (depends on semantics of domain), e.g. age between 0 and 150
- ❖ Restrictions depending only on relationship between values (functional dependencies)
- ❖ FD's arise naturally, e.g. for an entity set, an entity uniquely determines its attributes: so key \rightarrow attributes
- ❖ Note that we can't deduce FD's from an instance, they are "statements" about what instances are legal / allowed

Suppliers

- ❖ Suppliers(name, address, item, price)
 - ❖ name \rightarrow address
 - ❖ name, item \rightarrow price
- ❖ What other dependencies can we conclude
 - ❖ trivial ones: name \rightarrow name; name, item \rightarrow item
 - ❖ nontrivial: name, item \rightarrow address, price

closure of $F = F^+$

- ❖ Let F be a set of FD's
- ❖ F^+ is the set of FD's logically implied by F
- ❖ set of rules called "Armstrong's Axioms"
 - ❖ If $Y \subseteq X$ then, $X \rightarrow Y$ (reflexivity)
 - ❖ If $X \rightarrow Y$, then $XZ \rightarrow YZ$ (augmentation)
 - ❖ If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$ (transitivity)

Example

- ❖ $R(\text{City}, \text{State}, \text{Zip})$
 - ❖ FD1: $\text{City}, \text{State} \rightarrow \text{Zip}$
 - ❖ FD2: $\text{Zip} \rightarrow \text{City}$
- ❖ Does $\text{State}, \text{Zip} \rightarrow \text{City}, \text{State}, \text{Zip}$?
 1. $\text{Zip} \rightarrow \text{City}$ (given)
 2. $\text{State}, \text{Zip} \rightarrow \text{City}, \text{State}$ (from augmentation of above with State)
 3. $\text{City}, \text{State} \rightarrow \text{Zip}$ (given)
 4. $\text{City}, \text{State} \rightarrow \text{City}, \text{State}, \text{Zip}$ (from augmentation)
 5. $\text{State}, \text{Zip} \rightarrow \text{City}, \text{State}, \text{Zip}$ (transitivity with 2 and 4)

Attribute Closure X^+

- ❖ Computing and comparing F^+ is expensive
- ❖ Closure of Attributes, X , X^+ is set of all attributes A such that $X \rightarrow A$ is implied by the set of FD's
- ❖ So we can tell if $X \rightarrow Y$ is in F^+ by computing X^+ and checking if Y is in X^+

Computing Attribute Closure

- ❖ Illustrated by Example: Let $X=BD$, want to compute X^+
- ❖ FDs: $AB \rightarrow C$; $C \rightarrow A$; $BC \rightarrow D$; $ACD \rightarrow B$; $D \rightarrow EG$; $BE \rightarrow C$; $CG \rightarrow BD$; $CE \rightarrow AG$;
- ❖ $X(0) = BD$; $X(1) \{B,D,BD \text{ in lhs}\} = BDEG$; $X(2) = BDEGC$;
 $X(3) = BDEGCA$ (all attributes)

Equivalence of two FD sets

- ❖ F covers G (or G covers F) if $F^+ = G^+$
- ❖ minimal cover is when all FD's are of following nature
 - ❖ RHS has only one attribute
 - ❖ No FD $X \rightarrow A$ is unnecessary i.e. $(F - \{X \rightarrow A\})^+ \neq F^+$
 - ❖ No attribute in LHS is unnecessary, i.e. for $X \rightarrow A$, if Z is a proper subset of X then:
$$(F - \{X \rightarrow A\} \cup \{Z \rightarrow A\})^+ \neq F^+$$

Computing Minimal Cover

- ❖ Put FD's in standard form (i.e. one attribute in RHS)
- ❖ Minimize LHS of each FD (see if it can be removed while maintaining equivalence)
- ❖ Delete redundant FD's

Example: minimal cover of F

- ❖ $F = \{AB \rightarrow C, C \rightarrow A, BC \rightarrow D, ACD \rightarrow B, D \rightarrow EG, BE \rightarrow C, CG \rightarrow BD, CE \rightarrow AG\}$
- ❖ Step 1 results in: $AB \rightarrow C, C \rightarrow A, BC \rightarrow D, ACD \rightarrow B, D \rightarrow E, D \rightarrow G, BE \rightarrow C, CG \rightarrow B, CG \rightarrow D, CE \rightarrow A, CE \rightarrow G$
- ❖ Step 2: $ACD \rightarrow B$ simplifies to $CD \rightarrow B$ as $C \rightarrow A$
- ❖ Step 3: $CE \rightarrow A$ is redundant ($C \rightarrow A$), $CG \rightarrow B$ is redundant ($CG \rightarrow D, CD \rightarrow B$)
- ❖ Other results possible

Decomposition

- ❖ Lossless Join Decomposition of R into R1 and R2 if for every relation r of schema R the following is true
 - ❖ $r = \pi_{R1}(r) \bowtie \pi_{R2}(r)$
 - ❖ satisfied if $R1 \cap R2 \rightarrow R1$; or $R1 \cap R2 \rightarrow R2$
 - ❖ The original $R = R1 \bowtie R2$ (natural join of R1, R2)
- ❖ Dependency Preserving decomposition = when all FD's can be evaluated in the decomposed relations

Supplier Decomposition

- ❖ name \rightarrow address
- ❖ name, item \rightarrow price
- ❖ Decompose as:
 - ❖ SuppAddr(name, address)
 - ❖ SuppItemPrice(name, item, price)
- ❖ $R1 \cap R2 = \text{name} \rightarrow \text{name, address}$
- ❖ lossless decomposition

name	addr
S1	123 Any
S3	10 State
S2	1 Main
S4	11 State

name	item	price
S2	P3	100
S1	P2	20
S1	P1	10
S3	P4	1000
S2	P1	11
S4	P1	9

Examples

- ❖ Consider $R(\text{City, State, Zip})$, $R(C,S,Z)$ for short
- ❖ We have the FD's $CS \rightarrow Z$ and $Z \rightarrow C$
- ❖ We can decompose $R(C,S,Z)$ into $R_1(C,Z)$ and $R_2(S,Z)$ as $CZ \cap SZ \rightarrow C$ or $Z \rightarrow C$ (R_1)
- ❖ FD $Z \rightarrow C$ can be evaluated in R_1 but FD $CS \rightarrow Z$ can't be evaluated without a join
- ❖ Note that a decomposition can preserve dependencies but not be loss-less think $R(A,B,C,D)$ where $A \rightarrow B$ and $C \rightarrow D$ decomposed into $R_1(A,B)$ and $R_2(C,D)$

BCNF = Boyce-Codd NF

- ❖ Only non trivial FD's are where $\text{key} \rightarrow \text{attributes}$
- ❖ $R(C,S,Z)$ is not in BCNF because of FD: $Z \rightarrow C$ as Z is not a key of R
- ❖ A typical ER model is likely to be in BCNF, unless there are unexpected relationships between attributes
 - ❖ Are such FD's irrelevant? Do they tell us something that is not of much use? Is $Z \rightarrow C$ info of any use?

3NF

- ❖ Also allows that for FD: $X \rightarrow A$, A is part of a key
- ❖ So in $R(C,S,Z)$ which has keys CS and ZS , $Z \rightarrow C$ doesn't violate 3NF as C is part of key CS .

3NF violated

- ❖ If $X \rightarrow A$ violates 3NF, then either X is a proper subset of a key, i.e. part of a key determines an attribute (also called partial dependency)
- ❖ Or X is not a proper subset of any key; in that case $X \rightarrow A$ would be a transitive dependency when FD: $\text{key} \rightarrow X$ is considered (this is 2NF...)

Decomposing R into 3NF

- ❖ If we compute minimum cover then for each dependency $X \rightarrow A$ in minimum cover, we add the relation XA in the schema.
- ❖ Of course if dependencies are $X \rightarrow A$, $X \rightarrow B$, etc. then we combine them into a single relation, XAB
- ❖ This **set of relations** \cup **{key of R}** would give a 3NF decomposition of R

Normalization as discovering ERs

- ❖ One way to think of schema refinement is to discover entities and relationships from the data
 - ❖ A good / detailed ER model should end up directly as a 3NF or BCNF
- ❖ The functional dependencies guide us to determine entities and their relationships
- ❖ What if we already have a bunch of data, say in a universal relation?

Universal Relation

- ❖ A relation that captures all the information in schema
- ❖ Decomposable into smaller relations, but how?

dept	cid	course_name	sid	last_name	first_name	status	semester	year
CS	300	Programming II	2012144	Bush	George	4	2	2015
CS	300	Programming II	2014101	Obama	Barack	3	2	2015
CS	300	Programming II	2015001	Lincoln	Abraham	1	2	2015
CS	564	DBMS	2012144	Bush	George	4	1	2016
CS	564	DBMS	2014101	Obama	Barack	3	1	2016
CS	564	DBMS	2016001	Clinton	Bill	5	1	2016
Math	234	Calculus III	2012144	Bush	George	4	1	2013
PolSci	104	US Govt	2012144	Bush	George	4	1	2016
PolSci	104	US Govt	2012144	Bush	George	4	2	2015
PolSci	104	US Govt	2014101	Obama	Barack	3	1	2016
PolSci	104	US Govt	2015001	Lincoln	Abraham	1	1	2016

Discovering FD's

- ❖ could come from requirement of the data / application, i.e., constraint
- ❖ sometimes be inferred from data, but need validation
 - ❖ dept, cid -> course_name
 - ❖ CS,300 -> 'Programming II', etc.
 - ❖ sid -> last_name, first_name
 - ❖ sid -> status? or sid, semester, year -> status

In Practice

- ❖ Good to identify the functional dependencies in the data
- ❖ Use that to guide the logical schema (table definitions)
- ❖ 3NF is mostly good enough, and preserves FD's
- ❖ Surrogate keys can help with redundancy (with tradeoffs)
- ❖ Generally verify that there is not much redundancy
- ❖ Sometimes we may want to denormalize or not normalize it all the way to save on join costs

Multi-dimensional Data Model

Multi-Dimensional Data Model

25.2

- ❖ Think spreadsheets and reporting but generalized
- ❖ Metrics of interest vary across several dimensions
 - ❖ sales varying across product, stores, time
- ❖ The dimensions are arranged in hierarchies
 - ❖ store, city, region, country
 - ❖ date, week, month, quarter, year

A MDDB “Query” View

The screenshot shows the Microsoft Excel application window titled "Book1 - Microsoft Excel". The ribbon at the top includes tabs for Home, Insert, Page Layout, Formulas, Data, Review, View, Developer, Smart View Utilities, Smart View, and Smart Query. The Smart View tab is active, displaying options like Analyze, Designer Panel, Copy, Paste, Save, Open, Refresh, Automatic Refresh, Change Alias, Default, Indentation, Show Distinct Member Names, and Suppress Rows with No Data.

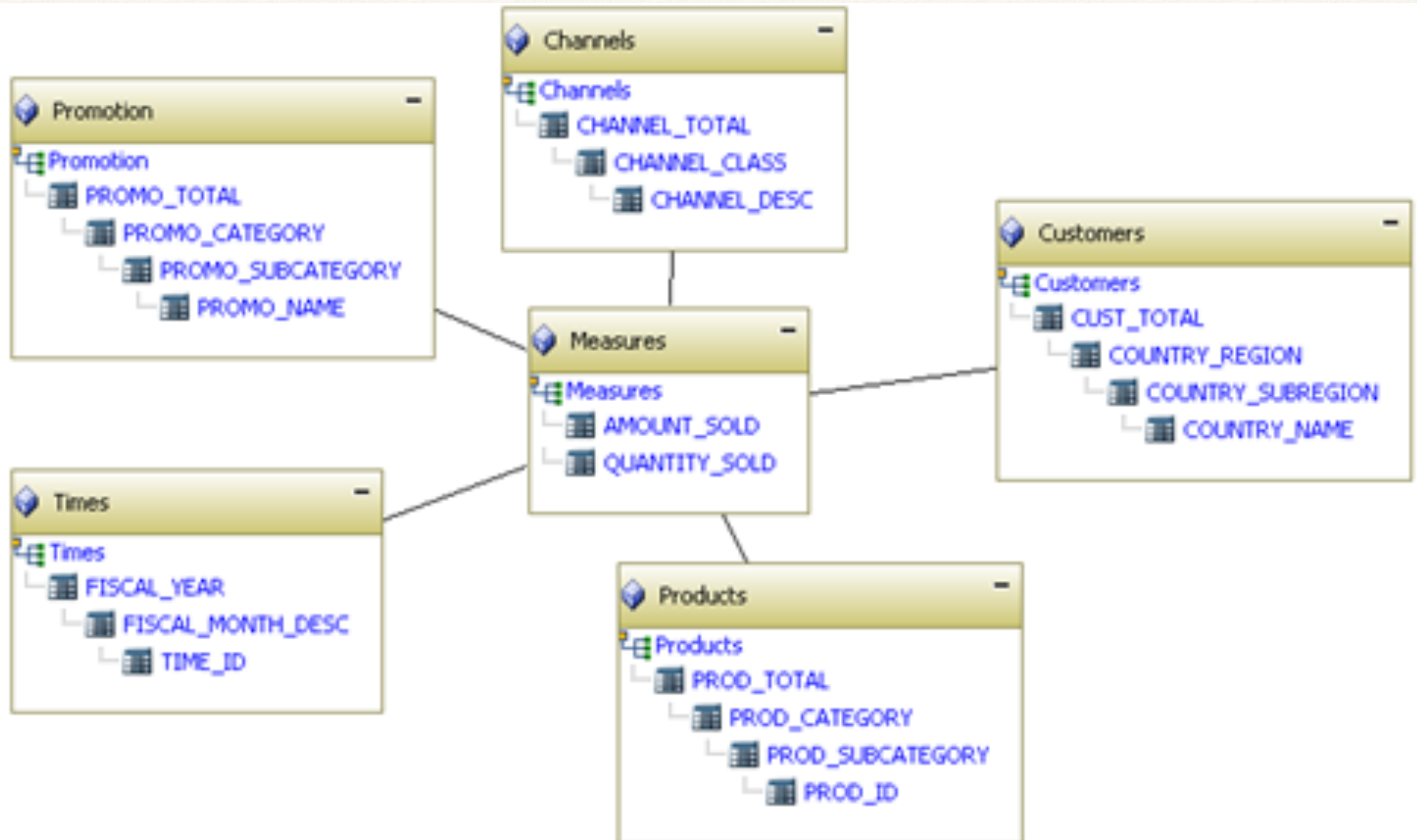
The main workspace displays a Smart View grid. The left sidebar contains several sections:

- Smart View**: Includes a search bar and a list of rows and columns.
- Rows**: A list of row labels including "Product", "Year", and "Measures".
- Columns**: A list of column labels including "Profit", "Margin", "Expenses", "Sales", and "COGS".
- Point of View**: A dropdown menu currently set to "Actual" and "New York".
- Sets for Market**: A section for market-related settings.
- Filters for Set Market (T)**: A section for filtering data based on the selected market.

The main grid displays data for various products across four quarters (Qtr1, Qtr2, Qtr3, Qtr4). The data is organized into groups for each quarter, with a "Total" row for each group. The products listed include Cola, Dark Cream, Birch Beer, Orange, Grape, Total Top 5 Products, Average Top 5, Old Fashioned, Vanilla Cream, Strawberry, Grape, Orange, Total Bottom 5 Products, and Average Bottom 5.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
		Qtr1	Qtr2					Qtr3					Qtr4				
		Total	Total	Total	Total	Total	Total	Total	Total	Total	Total	Total	Total	Total	Total	Total	Total
		Profit	Margin	Expenses	Sales	COGS	Profit	Margin	Expenses	Sales	COGS	Profit	Margin	Expenses	Sales	COGS	Profit
2	Cola	\$ 766	\$1,199	\$ 433	\$1,998	\$ 799	\$ 928	\$1,416	\$ 488	\$2,358	\$ 942	\$1,050	\$1,568	\$ 518	\$2,612	\$1,044	\$ 75
3	Dark Cream	\$ 383	\$ 761	\$ 378	\$1,491	\$ 730	\$ 762	\$1,162	\$ 400	\$1,958	\$ 796	\$ 709	\$1,131	\$ 422	\$1,987	\$ 856	\$ 64
4	Birch Beer	\$ 633	\$ 898	\$ 265	\$1,593	\$ 665	\$ 740	\$1,023	\$ 283	\$1,782	\$ 759	\$ 585	\$ 875	\$ 290	\$1,656	\$ 781	\$1,112
5	Orange	\$ 290	\$ 400	\$ 110	\$ 675	\$ 275	\$ 327	\$ 447	\$ 120	\$ 755	\$ 308	\$ 377	\$ 509	\$ 132	\$ 859	\$ 350	\$ 35
6	Grape	\$ 303	\$ 415	\$ 112	\$ 700	\$ 285	\$ 350	\$ 475	\$ 125	\$ 802	\$ 327	\$ 388	\$ 521	\$ 133	\$ 880	\$ 359	\$ 26
7	Total Top 5 Products	\$2,375	\$3,673	\$ 1,298	\$6,457	\$2,784	\$3,107	\$4,523	\$ 1,416	\$7,655	\$3,132	\$3,109	\$4,604	\$ 1,495	\$7,994	\$3,390	\$3,26
8	Average Top 5	\$ 475	\$ 735	\$ 260	\$1,291	\$ 557	\$ 621	\$ 905	\$ 283	\$1,531	\$ 626	\$ 622	\$ 921	\$ 299	\$1,599	\$ 678	\$ 64
9	Old Fashioned	\$ (583)	\$ (166)	\$ 417	\$ 185	\$ 351	\$ (636)	\$ (183)	\$ 453	\$ 207	\$ 390	\$ (747)	\$ (233)	\$ 514	\$ 223	\$ 456	\$ (62)
10	Vanilla Cream	\$ (352)	\$ (92)	\$ 260	\$ 542	\$ 634	\$ (379)	\$ (103)	\$ 276	\$ 585	\$ 688	\$ (706)	\$ (388)	\$ 318	\$ 434	\$ 822	\$ (51)
11	Strawberry	\$ 216	\$ 309	\$ 93	\$ 521	\$ 212	\$ 271	\$ 377	\$ 106	\$ 638	\$ 261	\$ 287	\$ 399	\$ 112	\$ 674	\$ 275	\$ 17
12	Grape	\$ 303	\$ 415	\$ 112	\$ 700	\$ 285	\$ 350	\$ 475	\$ 125	\$ 802	\$ 327	\$ 388	\$ 521	\$ 133	\$ 880	\$ 359	\$ 26
13	Orange	\$ 290	\$ 400	\$ 110	\$ 675	\$ 275	\$ 327	\$ 447	\$ 120	\$ 755	\$ 308	\$ 377	\$ 509	\$ 132	\$ 859	\$ 350	\$ 35
14	Total Bottom 5 Products	\$ (126)	\$ 866	\$ 992	\$2,623	\$1,757	\$ (67)	\$1,013	\$ 1,080	\$2,987	\$1,974	\$ (401)	\$ 808	\$ 1,209	\$3,070	\$2,262	\$ (26)
15	Average Bottom 5	\$ (25)	\$ 173	\$ 198	\$ 525	\$ 351	\$ (13)	\$ 203	\$ 216	\$ 597	\$ 395	\$ (80)	\$ 162	\$ 242	\$ 614	\$ 452	\$ (5)

A MDDB Schema



MDDDB & Relational: aka Star Schema

- ❖ MD Schema can be modeled as a Relational Schema
- ❖ Relational is flexible and powerful
- ❖ Fact table captures the measures
- ❖ Dimension tables capture the dimensions, the hierarchies being implicit in the column names and rollup dependencies

A Star Schema

- ❖ Fact

- ❖ measures(channel_id, country_id, prod_id, time_id, promo_id, amount_sold, quantity_sold)

- ❖ Dimensions

- ❖ channels(channel_id, channel_desc, channel_class, channel_total)
 - ❖ customers(country_id, country_name, country_subregion, country_region, cust_total)
 - ❖ products(prod_id, prod_subcategory, prod_category, prod_total)
 - ❖ times(time_id, fiscal_month_desc, fiscal_year)
 - ❖ promotion(promo_id, promo_name, promo_subcategory, promo_category, promo_total)

Levels of Abstraction in DBMS

1.5.2

- ❖ Physical Schema is the way the relations are actually stored in SSD / HDD. It also defines indexes, statistics etc. defined on the table. (Indexes are defined using DDL.)
- ❖ Logical (Conceptual) Schema is the DDL for creating the table. (It can sometimes specify the physical schema.)
- ❖ External Schema is the DDL that define's the external user's view of the database. Though some "base" tables are directly visible, most of it is protected through views.

Views

- ❖ Views look like tables to user but can be defined by pretty much any SQL query
- ❖ Only the definition stored in the database
- ❖ Simple views on a single table can be used for insert/delete/update operations as the operation is reflected on the underlying base table.

Views

```
create view Hourly as
  select ssn, name, lot, wages, hours
  from Employees
 where type = 'H';
```

```
create view Contract as
  select ssn, name, lot, contracted
  from Employees
 where type = 'C';
```

```
create view Regular as
  select ssn, name, lot
  from Employees
 where type = 'R';
```

Class Hierarchies using Views

Contract

ssn	name	lot	contractid
301	Bill	1	1001

Hourly

ssn	name	lot	wages	hours
201	George	2	15	2
202	James	2	20	40

Regular

ssn	name	lot
101	Abe	1

Employees

ssn	name	lot	type	wages	hours	contractid
201	George	2	H	15	2	null
202	James	2	H	20	40	null
301	Bill	1	C	null	null	1001
101	Abe	1	R	null	null	null

Views

- ❖ Hide complexity
- ❖ Limited Access / Security
- ❖ External Schema can be independent of Conceptual Schema (definitions of base tables)
 - ❖ logical data independence
 - ❖ can change conceptual schema without affecting users

Steps in Database Modeling

- ❖ Both data that's there and queries that may be asked
- ❖ ER may be useful for high level
- ❖ Concrete: Tables, Attributes
- ❖ Refine: Establishing relationships between attributes and tables
- ❖ Data Definition and Index structures
- ❖ Additional index structures

