

Normalization



Normalization

- Tables are important, but properly designing them is even more important so the DBMS can do its job
- **Normalization** → the process for evaluating and correcting table structures to minimize data redundancies
 - Reduces the likelihood of *anomalies*
- A process that goes by levels, refining the design each time
- Although we deal with tables, the refinement is mostly on the relationship level

Just do it

- So what's the point?
 - Each level reduces more redundancy
 - Each level also makes DB operations slower
- There are many levels of normalization, but the typical stopping point is the 3rd (3NF)

Case Study

- Consider a construction company that handles several building projects
- Manage projects, clients, billing, and employees

A First Try – Good? Bad?

FIGURE
5.1

Tabular representation of the report format

Table name: RPT_FORMAT

Database name: Ch05_ConstructCo

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	June E. Arbough	Elect. Engineer	84.50	23.8
		101	John G. News	Database Designer	105.00	19.4
		105	Alice K. Johnson *	Database Designer	105.00	35.7
		106	William Smithfield	Programmer	35.75	12.6
		102	David H. Senior	Systems Analyst	96.75	23.8
18	Amber Wave	114	Annelise Jones	Applications Designer	48.10	24.6
		118	James J. Frommer	General Support	18.36	45.3
		104	Anne K. Ramoras *	Systems Analyst	96.75	32.4
		112	Darlene M. Smithson	DSS Analyst	45.95	44.0
22	Rolling Tide	105	Alice K. Johnson	Database Designer	105.00	64.7
		104	Anne K. Ramoras	Systems Analyst	96.75	48.4
		113	Delbert K. Joenbrood *	Applications Designer	48.10	23.6
		111	Geoff B. Wabash	Clerical Support	26.87	22.0
		106	William Smithfield	Programmer	35.75	12.8
25	Starflight	107	Maria D. Alonzo	Programmer	35.75	24.6
		115	Travis B. Bawang	Systems Analyst	96.75	45.8
		101	John G. News *	Database Designer	105.00	56.3
		114	Annelise Jones	Applications Designer	48.10	33.1
		108	Ralph B. Washington	Systems Analyst	96.75	23.6
		118	James J. Frommer	General Support	18.36	30.5
		112	Darlene M. Smithson	DSS Analyst	45.95	41.4

Observations – The Good

- "Good"
 - An employee is only listed in a project once, so PROJ_NUM with EMP_NUM can give you job classification
 - Hourly wage and hours worked are included
 - Total charge for an employee and project as a whole can be calculated
- Neutral
 - An employee can work on more than one project

The Bad

- Nulls! Especially in the expected implied PK of PROJ_NUM
- Easy for data inconsistencies and anomalies occur, especially if relying on human entry (ex. abbreviations)
- Data redundancies yield inconsistencies
 - Updating a particular employee can require many updates
 - Inserting new employees without a project is trouble
 - Project and Employee data are mixed, so if an employee leaves data gets deleted

The Ugly

- Reporting on this table can yield different results depending on the anomaly
 - Think of the job abbreviation thing again... if you want a report of all hours worked by a job class how do you do it if they are inconsistent?
- Data entry is rough even if you audit the errors mentioned already
 - A lot of data is repeated and people are poor at that

Goals of Normalization

- A table represents only a single object and its attributes
- Data should be kept in one table where possible (controlled redundancy) to eliminate update anomalies
- All data in a row must depend on the primary key and nothing else to ensure PKs uniquely identify rows
- No update, insertion, or deletion anomalies so integrity and consistency is ensured

Functional Dependency Ahhh!

- Functional Dependency \rightarrow Attr B is functionally dependent on attr A if each value of A determines one and only one value of B
 - Can also be thought of as "A determines B"
 - License # is fully functionally dependent on SS#
 - Name is not
- Fully Functional Dependency with a composite key (the confusing one) \rightarrow B is functionally dependent on A but not any subset of A

1NF Step 1

- Step 1 – Eliminate Repeating Groups
- **Repeating Group** → multiple entries exist for the same key
 - A column cannot contain more than one data item
→ *Repeating within columns*
 - Cannot be more than one column for the same data
→ *Repeating across columns*
 - A value cannot be assumed to span into multiple rows (re: cascading down into nulls)
→ *Repeating across rows*

1NF Step 1

- Row repetition can be eliminated by filling in the null values and assuring no rows have exactly the same data
- Column repetition is eliminated by the creation of a second table whose PK is composed of the original table's PK and the column in question (have to do this after step 2...)

1NF Step 2

- Identify the minimal PK that will uniquely identify each row
- All other attributes should thus depend on the PK columns

1NF Step 3

- Identify all dependencies between columns
 - All attributes should be dependent on the key
 - Some attributes may be dependent on only part of the key (*partial dependency*)
 - Some attributes may be dependent on others (*transitive dependency*)
- Best way to do this is probably a dependency diagram (pg 160)

1NF Requirements

- Unique PK identified and all non-PK attributes depend on it
 - Implies no duplicate rows
- No ordering of rows nor columns
- Each row-column intersection contains one and only one value in the domain
 - No nulls!
 - Some do not feel this is a requirement
- These last two combine to mean no repeating groups

2NF

- If a PK is a single attribute then it is already in 2NF, the below process only applies to composite primary keys
- Write each component of the key on a separate line with the last line the entire key
- Figure out which attributes are dependent on which parts
- Create new tables from each of these lines

2NF Requirements

- Already in 1NF
- No partial dependencies

3NF

- 2NF eliminates partial dependencies but not transitives, which can still leave anomalies, 3NF eliminates these
- Identify determinants
 - An attribute whose value determines another
 - There will be one per transitive dependency
- Identify dependent(s) of the determinant(s)
 - Ex. student status → charge per credit hour
- Remove the dependencies and place in table with the determinant as the PK
 - Leave the determinant as a FK in its original table

3NF Requirements

- 2NF
- No transitive dependencies

More NF?

- The book mentions BCNF and 4NF but these are not really used in practice except very specific cases
- For BCNF, most 3NF tables will conform to its requirements anyway

Design Decisions

- Normalization refines relationships and eliminates redundancies
- It ***does not*** create good designs

Primary Keys

- Sometimes natural keys are hard to deal with
 - If it is JOB_NAME for example, it is easy for someone to type it incorrectly in other tables where it is an FK
- In this case (where PK is an FK) it may be better to use a ***surrogate key***
- What's the problem with introducing a surrogate key to a system that is already 3NF?

Naming Conventions

- Naming conventions will probably fall into one of two situations:
 - Business rules specified
 - You get to choose
- If you are in a position to pick them, make sure they are consistent and informative

Are all attributes accounted for?

- After designing and normalizing make sure all the information the database needs is present or can be calculated
- If it isn't you must add new columns and possibly run through the normalization process again

All relationships?

- Same thing as with attributes

Atomicity

- **Atomicity** → whether an attribute can be broken down into more than one sensible attribute
 - Address → "123 Fake St, Springfield, IL" → **not atomic**
 - Address → "123 Fake St", City → "Springfield", State → "IL" → **is atomic**
- Atomicity is a bit fuzzy and is often considered to be a "means nothing" sort of term in its non-adjusted definition (the above is my adjusted version)

Granularity Refinement

- ***Granularity*** is the level of detail of data
 - The *most* granular you can get it atomic
- Typically you are concerned with PKs here, but not always
- For a student, are credit hours specified as a total or only for the current semester?
- What about an employee and how many hours they work on a project?

Historical Records

- For the past slide, if choosing the "current time period" option, you need some way to maintain a history of the data for lookups of past periods
- For example, what if a person's consulting rate changes over time? A properly designed table can handle this

Derived Attributes

- Consider whether it is better to store these or to calculate them on the fly
- How will end users be using the data?
- What kind of performance is needed and what is the hardware capable of?

Surrogate Keys

- Like we discussed in the last lecture, sometimes a composite key is too hard to maintain
 - When it is a FK it is probably not worth using
- In these cases we create a surrogate key to serve as a PK
- Single column, numeric, automatically incremented by the database

Be careful...

- Surrogate keys can lead to similar/duplicate row entries, which are undesirable
- Must enforce "unique" constraints on other attributes to ensure this doesn't happen
- This may be hard, such as the case of names
 - An externally defined attribute would be better in this case

TABLE
5.4

Duplicate Entries in the Job Table

JOB_CODE	JOB_DESCRIPTION	JOB_CHG_HOUR
511	Programmer	\$35.75
512	Programmer	\$35.75

Design & Normalization

- Normalization is part of the design process
- As you are ER modeling you should also be normalizing
- Normalization is an intense analysis of a specific entity and its relationships. It reduces redundancy and thus anomalies
- Recommendation: get a rough sketch of the ER complete and then use normalization to refine parts of it individually
- Section 5.7 gives a walkthrough of a design process

Denormalization

- Normalization is not a one-size-fits-all process
- Occasionally design decisions are made that break normalization either for ease of use or speed considerations
 - Is storing both zip codes and cities redundant?
 - The "Evaluate PK Assignments" section talks about a situation where having a transitive dependency is acceptable
 - Table 5.6

Homework

- Review Questions → None, but again good review
- Problems → 1, 2, 3, 4, 8, 9, 10