

Database design in agile projects

Heli Helskyaho
APEX Connect 2017

Introduction, Heli

- * Graduated from University of Helsinki (Master of Science, computer science), currently a doctoral student (Big Data, Unified Databases, Multi-model Databases) at University of Helsinki
- * Worked with Oracle products since 1993, worked for IT since 1990
- * Data and Database!
- * CEO for Miracle Finland Oy
- * Oracle ACE Director
- * Ambassador/EOUC (EMEA Oracle Users Group Community)
- * Public speaker and an author
- * Winner of Devvy for Database Design Category, 2015
- * Author of the book Oracle SQL Developer Data Modeler for Database Design Mastery (Oracle Press, 2015), co-author for Real World SQL and PL/SQL: Advice from the Experts (Oracle Press, 2016)



ORACLE



ORACLE

Oracle SQL Developer Data Modeler for Database Design Mastery

Design, Deploy, and Maintain World-Class Databases
on Any Platform

Heli Helskyaho
Oracle ACE Director

Forewords by C.J. Date and Tom Kyte



Oracle
Press

Real World SQL & PL/SQL

Advice from the Experts

Arup Nanda
Brendan Tierney
Heli Helskyaho
Martin Widlake
Alex Nuijten



Oracle
Press

Introduction, Topic

- * Why do we still design the database? Why is designing the application not enough?
- * What is database design?
- * How to design a database on agile systems development process? Why is it different than before? How is it different than before?
- * Why does agile often fail?

Why to design?

- * ”Data is the most valuable property in our company”
- * ”Why do we need to design the database? We already design the application!”

Why is designing the application not enough?

- * Point of view
- * First increment vs. 20 years from now
- * "the whole picture"
- * Same terminology, different meaning
-> misunderstandings
- * ...

Example

- * 1. iteration: ORDER
- * Info we have: layout for the user interface, very general use case (analysis) on making an order
- * So let's model....

ORDER

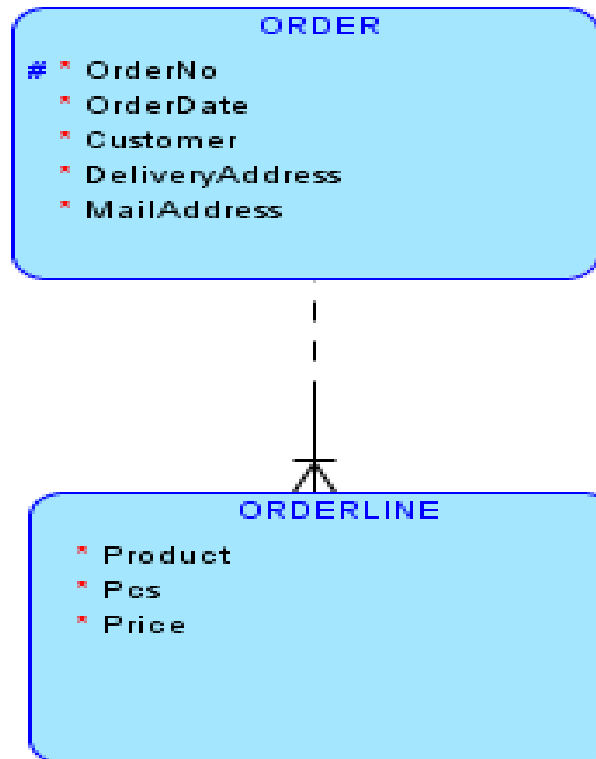
Order#	
Order Date	

Customer (name)	
Delivery address	
Zip code and city	

ORDER LINE

Product	Pcs	Price	Total

Make an Order



What went wrong?

- * Where can we find the **customer** for the order? Who cares. The use case "Create a Customer" is on the 2. iteration. And exactly the same with the **product** information.
- * So we have modeled the customer info right to the order -> duplicated information, no references but attributes,... And the same thing with the product information.
- * How do we know what a certain customer has ordered?
- * How do we know who has ordered certain product?
- * We have no way to produce any statistics...but nobody (=application designer, project manager,...) cares because statistics will be in iteration 7. ***We will think about that later.***

What is important?

- * It is very important to **plan the iterations and increments well** and **understand where and how much they overlap**. The less they overlap the better.
- * Always start with the most **difficult** and the most **centric** things, do not start with the easy things.
- * Always **ask questions** if you have a feeling that something is not right.

Where to aim?

- * A total mistake in modeling is difficult and expensive to fix later (at least the 20/80 rule...).
- * When designing the database you must understand **the big picture**: concepts and how they are connected to each others (entities and relationships)
- * GOAL 1: **Data integrity and quality.**
- * GOAL 2: How to **save and retrieve data.**

What is database design?

- * 4 (5) phases, over and over again
 - * Requirement analysis
 - * Conceptual design
 - * Logical design
 - * Physical design
 - * (Transaction design)

Designing the database

1. **Requirement analysis:** finding and analysing the requirements the future end users have

Result: specification of user requirements

- **data** requirements
- **functional** requirements

Also requirements for security, performance, ...

Designing the database

2. **Conceptual design.** "Interpretation" of all the requirements to a formal presentation (conceptual model).

Result: conceptual schema, also textual documentation is possible/recommended (to make sure all the knowledge is documented)

This is a tool for communication with end users.

Designing the database

3. **Logical design:** transforming the conceptual model into a logical data model and a logical schema that the RDMS understands

Result: relational-database schema
(relational schemas and constraints)

Designing the database

4. **Physical design:** instances, tablespaces, indexes, disks ...

And all of these phases over and over again...
(this is different, we have always done that but not so many times and in such a short cycles)

Designing the database

Might also include (depending who you ask)

- * **5. Transaction design:** transforming the functional descriptions in conceptual model to operations that will handle the database queries, transactions.

Designing the database, a bit closely about ...

- * Conceptual design
- * Logical design
- * Physical design

Conceptual design

- * Main idea: saving the data and retrieving it -> **DATA**
- * Information needed: everything you can find
- * Requirements and analysis usually half way -> need a lot of questions and answers
- * Interview the end users! Officially, unofficially
- * Model only the **target**, not the whole world

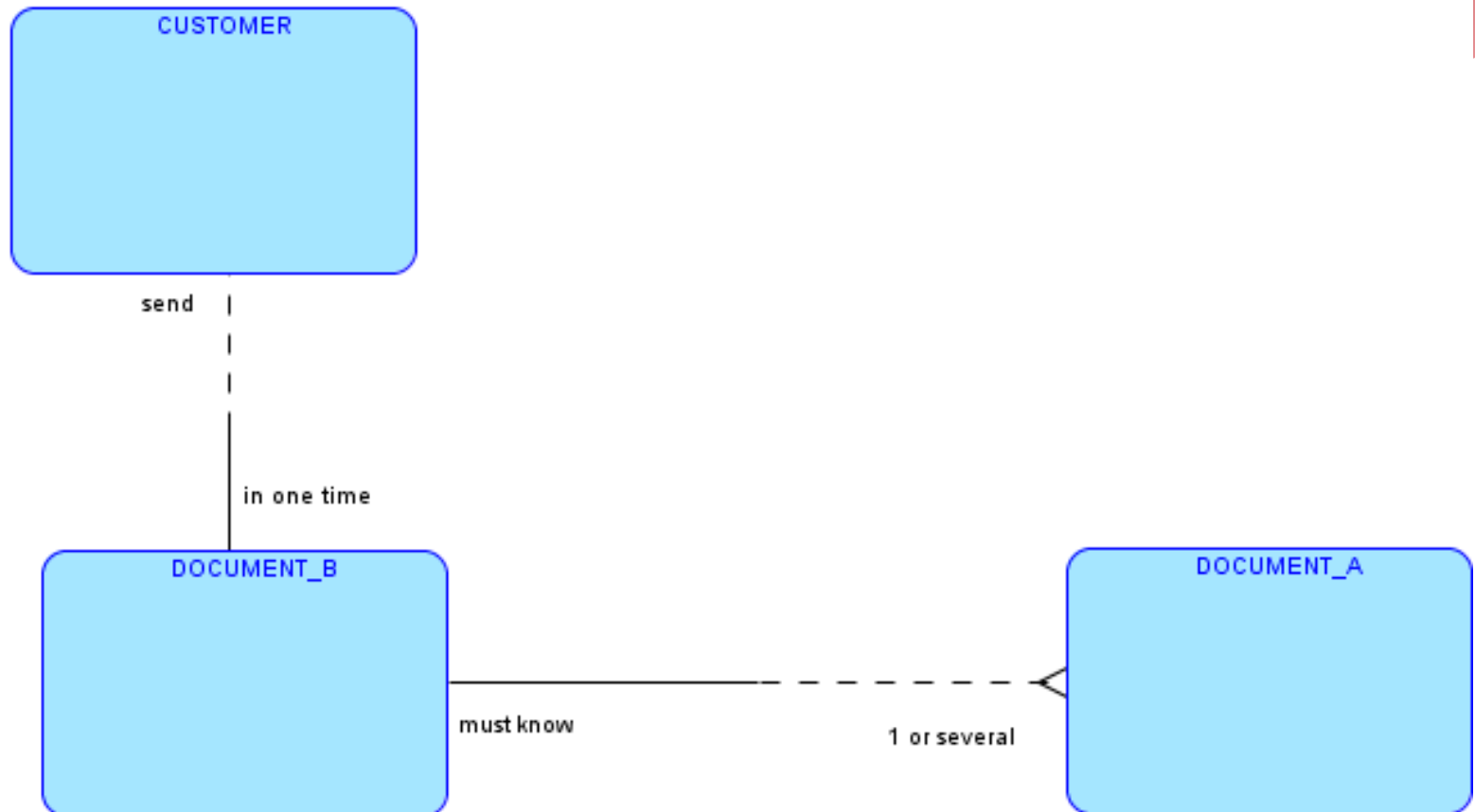
Conceptual design

- * Use right **terminology** and clear names, much easier to communicate with the end users (one of the reasons to model!)
- * Try to find and understand the **main** concepts and their relationships (these are the most difficult to change during the iterations)

Conceptual design, an example

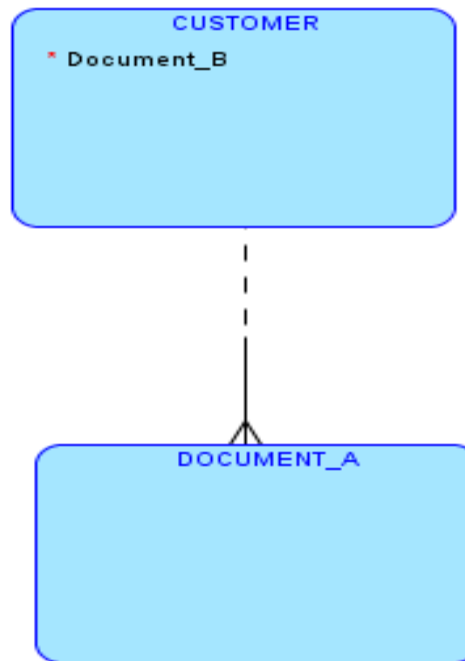
- * ”At one time one will make 1 or more Document/-s A and exactly one Document B for a Customer. One must know exactly which Document B was with which Document/-s A.”
- * EASY AND CLEAR!

Conceptual design, an example



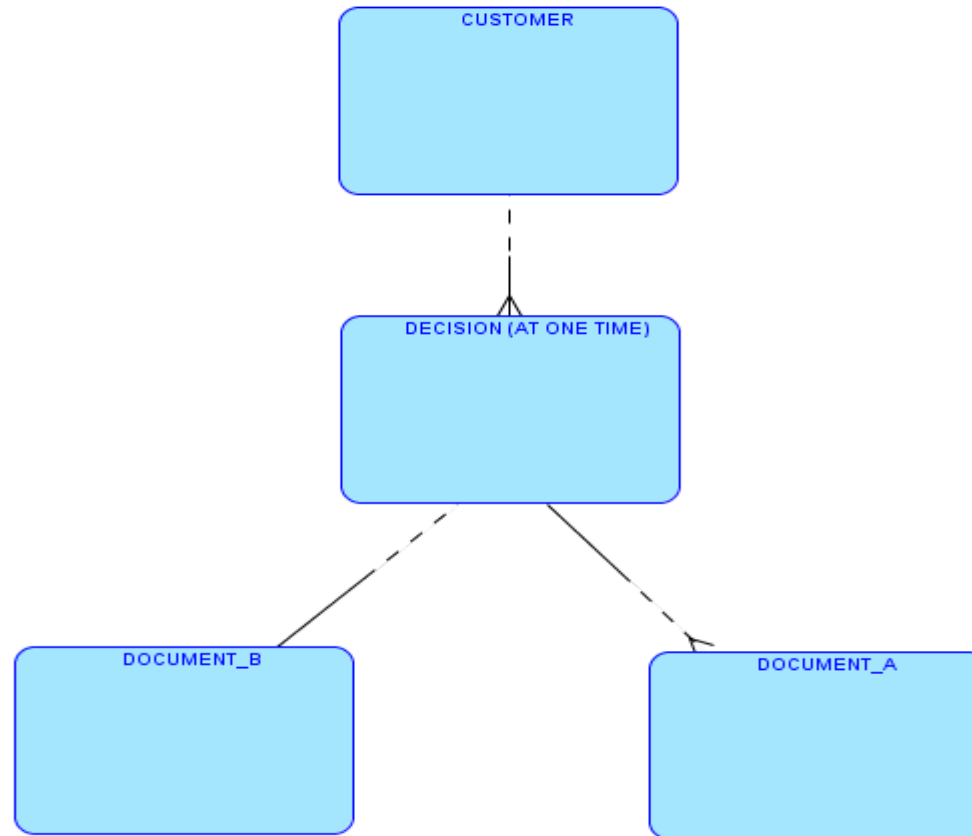
Conceptual design, an example

* OR?



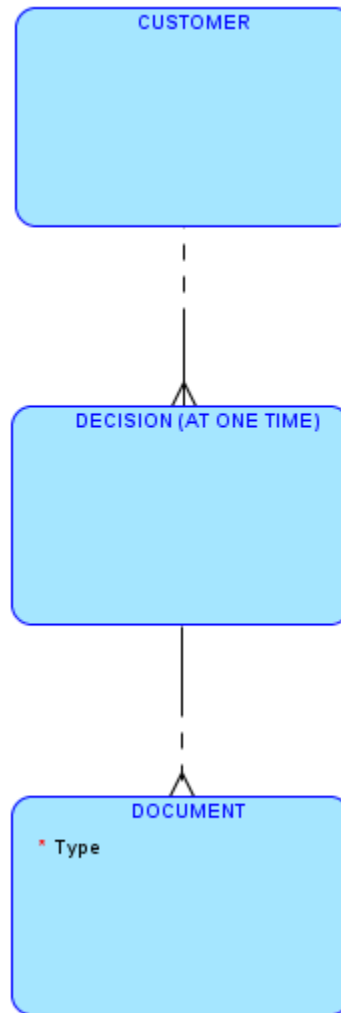
Conceptual design, an example

* OR?



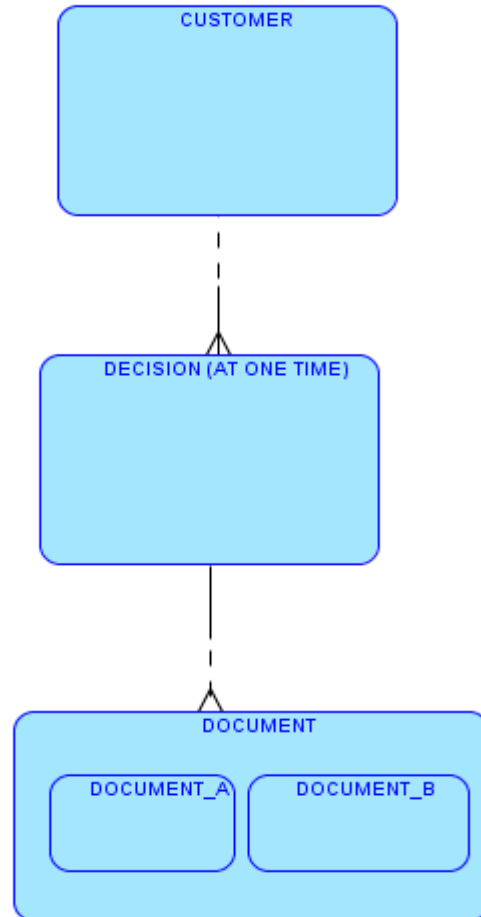
Conceptual design, an example

* OR?



Conceptual design, an example

* OR?



Conceptual design

- * Or one of the other dozen different ways to model this requirement?

Conceptual design

- * Modeling is **difficult** because
 - * Spoken/written language is not exact
 - * Usually all the "important" things are those that "everybody knows" so they are not told.
 - * At this stage we do not know one important thing: how the data will be **retrieved**? That will be on iteration 9...
- * Modeling is **mandatory** because when modeling the database designer realizes **what must be asked!**

Conceptual design, principles

- * Everybody has their own and preferred ways of modeling (“handwriting”) and that is ok, but it is important to be **consistent** at least inside one model or system or application.
- * Naming standards
- * Documentation
- * Comments from colleagues
- * Reviews, audits,...
- * ...

Logical design

- * Start with the conceptual model
- * **Specify and sharpen**, make **decisions** on more difficult modeling issues
- * Now you should already have the information how the database is supposed to be used, functional descriptions (usually you do not have it, not all at least 😞)

Logical design, Transformation

Two different kind of transformations
(in both the idea is conceptual model -> logical model):

1. Transformation from **notation** to another

- * UML -> ER
- * ER -> UML
- * ER and UML

2. Transformation from conceptual ER to logical ER
(same notation)

Transformation from notation to another

- * Transformation is quite simple from ER to UML or from UML to ER (can even be automated) but...

Difference between ER and UML

- * UML: both data requirements and functional requirements (not all the data requirements though)
- * ER: data requirements
- * Notation is not the issue but **what** are you trying to model. The **point of view** and the **need**.
- * **Is it possible in just one "picture" to model both data requirements and functional requirements? All of them?**

ER, UML?

- * I would recommend, ER for data requirements and UML for functional requirements
- * Then check the UML model against the ER model and make sure all data requirements that should have been implemented at the UML model are there (there is a meaning for them)
- * I would recommend to do it manually, so you can also review the ER model at the same time.
- * Try to test the models really work.

ER, UML?

- * Remember that it is possible that the class model has information that the data model does not have
 - * Age etc.
- * Also data model can have information that class model does not
 - * Creator, created (triggers, UI knows nothing about it)
- * So the models are not necessary identical!

Transformation from conceptual ER to logical ER (same notation)

- * If you have a tool, just press the button...

Normalization

Always the main principle and with a reason!

- * All the attributes are **atomic**. And the data has been saved only **once**.
- * **Why?** To make sure the data is good quality and correct.
- * Optimising by not normalising? OK, but you are now getting data which is not necessary good quality...

Other issues

- * Database privileges
- * History data
- * Cleaning the database
- * Documentation
- * ...

Physical design

- * In short:
 - * Start with the logical model
 - * Design the disc usage, count the need for the space etc.
 - * Indexes
 - * Schemas
 - * Implementations to all objects (tablespace, storage, user rights,...)
 - * DDLs

Physical design

- * How, who and where to document?
- * How to document any changes (patch, changes in database objects, ...)?
- * Backup and recovery (test that it works every now and then!)

Agile database design, in short

- * Agile makes database design harder than ever. Long experience in different projects is most valuable so you might be able to predict some problems beforehand.
- * **Agile is no excuse for not designing!**
- * **All the rules for database design still apply!**

Agile database design, in short

- * Agility costs. It is very important to analyze everything well so you know what is the price tag of your decision. (short term, long term)
- * Maintenance!
- * The more agile the more you must design and analyze. And unfortunately prioritize...

A tool

- * To be able to be agile or at least flexible one needs a tool for database design.
- * From Oracle the solution is Data Modeler (part of the SQL Developer)
- * To have less complains about the changes in database people testing needs also tools to be able to change their test material and testcases...

Oracle SQL Developer Data Modeler

- * To be efficient in designing you need a tool: my recommendation is Data Modeler
 - * Free of charge
 - * Support for many different databases
 - * Support for both documenting the existing databases and designing a new one (and maintaining that)
 - * Support for reporting, naming standards, glossaries, design rules, ...
 - * Support for version control and multiuser environment
 - * Support for everything you need for database design plus more

Performance

- * Plan, count, ...
- * Test! With a real material (size, quality,...) and the right versions of program.

Tuning

- * Hardware and operating system
- * DBMS
- * Model level (logical, physical, transaction)
- * All three levels affect each others

Indexing

- * Select **might** be faster
- * Insert, Update, Delete slower
- * Not a silver bullet

Tuning the queries

- * Transform the query to a more efficient form (**execution plan**) with the same outcome (**result**) than the original query (**equivalent**).

Tuning, the process

- * Ask direct questions. (“Nothing is working” is not an answer)
- * Be systematic and find the real problem
- * Solve the problem
- * Document the problem and the solution (new problems because of the solution?)

Refactoring the database

- * Analyze the change from the **database** perspective
 - * The phase of the database, development vs production
 - * The phase of usage of the database, development vs production
 - * Completely new database vs "old" database
- * Analyze the change from the **project** perspective (cheep coding now is not an excuse for a bad database design)

Refactoring the database

- * Analyze the change from the **maintenance** point of view (would it be more clear to call discount DISCOUNT than AMOUNT?)
- * Versioning
- * Always remember somebody will maintain this system and this database
- * Always remember clear and understandable solutions, do not try to be too clever

Refactoring the database

- * The problems related to refactoring the database usually have nothing to do with the database but everything to do "on top" of the database
- * Refactoring the database is nothing special, just normal database design process

Why does agile fail?

- * Waterfall is just called agile ☹️
- * Planning fails
- * Communication fails
- * Programming start on the first iteration.
 - * First n incrementations should be just the architecture and the database, before event trying to do anything else
- * The goal changes, new tasks on the list
- * Not enough testing, testing too late
- * ...

Conclusions

- * If you think you have/will have valuable data in your database, you **MUST** design the database.
- * The database must be designed by a person who understands how the database works and knows how the **DATA** should be modeled.
- * The database can not be designed alone in your room, you need all the different experts to give you the right info for making decisions.

Conclusions

- * The database must be designed and for the right purpose .
- * Data integrity and data quality must be the guiding rules.
- * All that you learned is school about database design still applies.
- * More communication! Respect the others, they know what they are doing.

Conclusions

- * If the timetable is tight you need more and more thinking and designing to be sure you are making the right decisions to be sure you will not need to change everything everytime.
- * Prioritization! For everybody (even the end users). Nothing is more stressful than too much work in no order and too little time.
- * Team spirit will help you through anything! Positive attitude!

THANK YOU!

QUESTIONS?

Email: heli@miracleoy.fi

Twitter: [@HeliFromFinland](https://twitter.com/HeliFromFinland)

Blog: Helifromfinland.com