CERN School of Computing

Fundamentals of Database Design

Zornitsa Zaharieva

CERN

Data Management Section - Controls Group

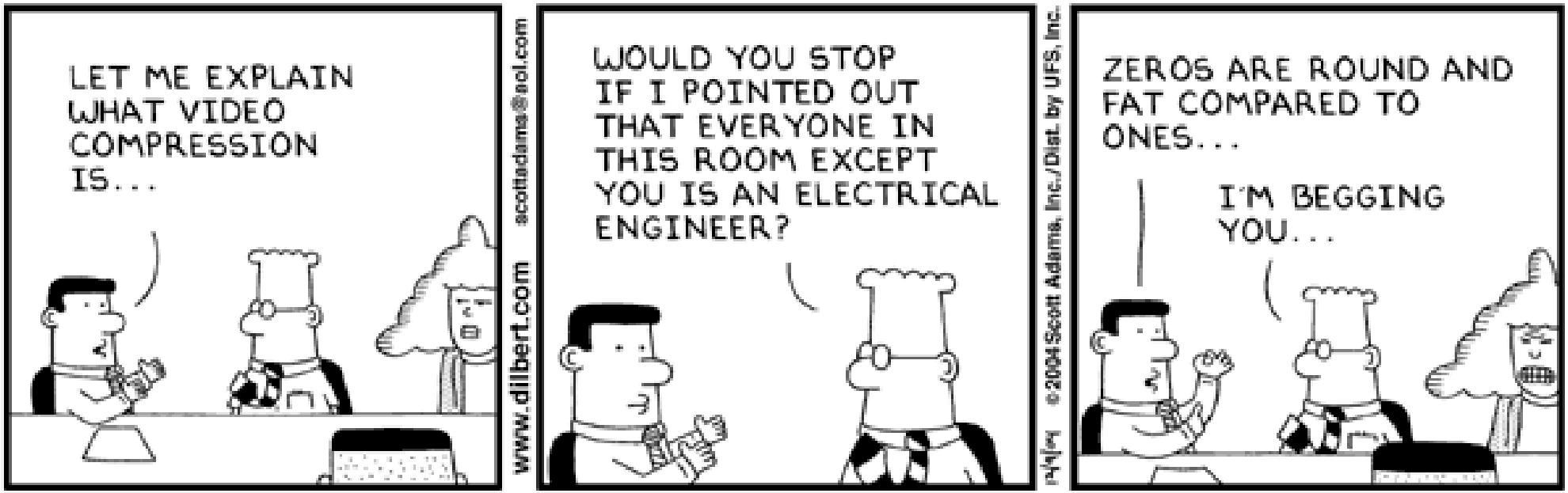
Accelerators and Beams Department

/AB-CO-DM/

23-FEB-2005

CERN School of Computing

Contents


LET ME EXPLAIN WHAT VIDEO COMPRESSION IS...
www.dilbert.com scottadams@aol.com
WOULD YOU STOP IF I POINTED OUT THAT EVERYONE IN THIS ROOM EXCEPT YOU IS AN ELECTRICAL ENGINEER?
© 2004 Scott Adams, Inc./Dist. by UFS, Inc.
ZEROS ARE ROUND AND FAT COMPARED TO ONES...
I'M BEGGING YOU...

CERN School of Computing

Databases - Evolution

- Data stored in file systems – problems with
  : redundancy
  : maintenance
  : security
  : efficient access to the data
- Database Management Systems
  Software tools that enable the management (definition, creation, maintenance and use) of *large amounts* of *interrelated data* stored in a computer accessible media.
- 1st generation of Database Management Systems
  : based on hierarchical and network models
- 2nd generation of DBMS
  : 1969 Dr. Codd proposed the relational model

CERN School of Computing

Capabilities of a Database Management System

- Manage persistent data
- Access large amounts of data efficiently
- Support for at least one data model
- Support for certain high-level language that allow the user to define the structure of the data, access data, and manipulate data
- Transaction management – the capability to provide correct, concurrent access to the database by many users at once
- Access control – the ability to limit access to data by unauthorized users, and the ability to check the validity of data
- Resiliency – the ability to recover from system failures without losing data

CERN School of Computing

Data Model

- A mathematical abstraction (formalism) through which the user can view the data

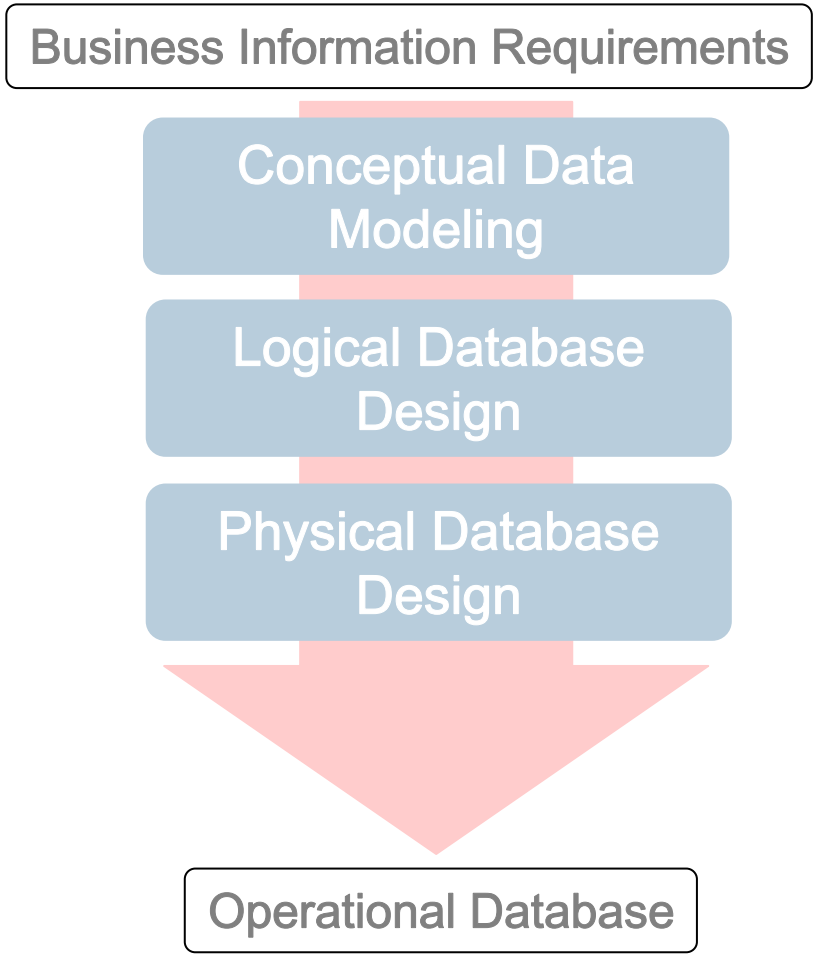
- Has two parts
  1. A notation for describing data
  2. A set of operations used to manipulate that data

- Examples of data models
  : relational model
  : network model
  : hierarchical model
  : object model

CERN School of Computing

Design Phases

- Difficulties in designing the DB's effectively brought design methodologies based on data models
- Database development process



Conceptual Design

Produces the initial model of the real world in a conceptual model

Logical Design

Consists of transforming the conceptual schema into the data model supported by the DBMS

Physical Design

Aims at improving the performance of the final system

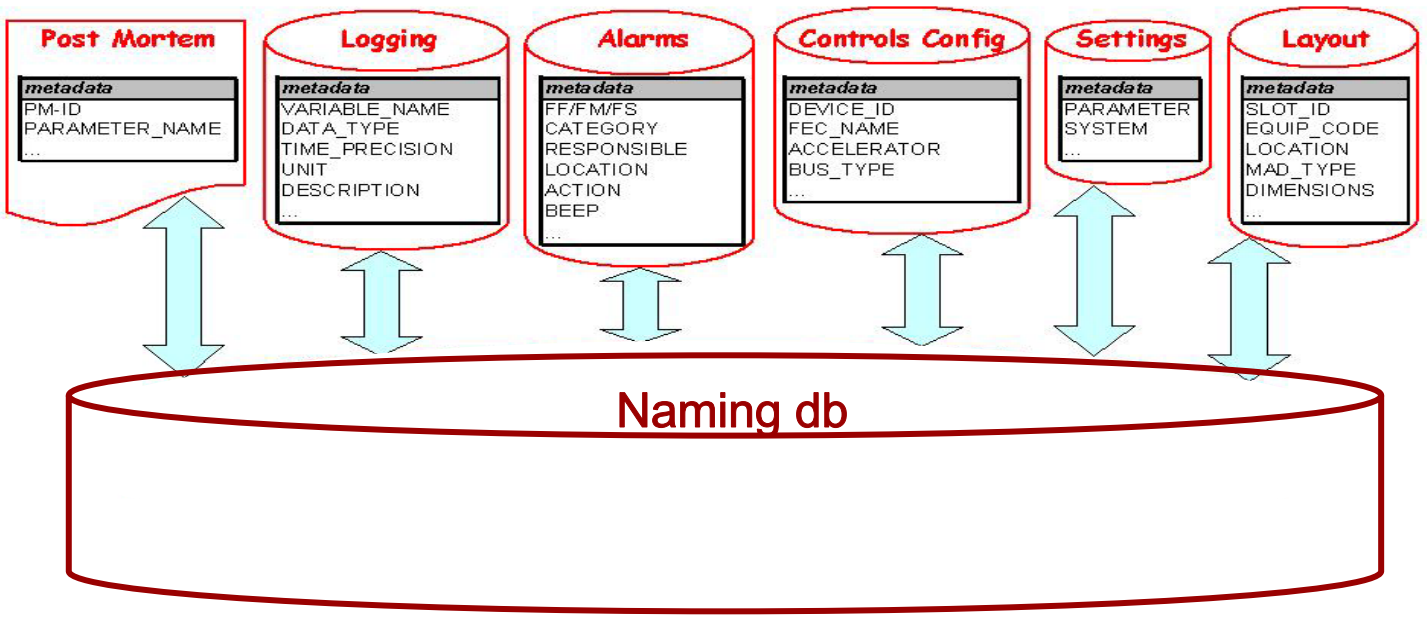
Conceptual Design

- The process of constructing a model of the information used in an enterprise
- Is a conceptual representation of the data structures
- Is independent of all physical considerations
- Should be simple enough to communicate with the end user
- Should be detailed enough to create the physical structure

CERN School of Computing

Information Requirements – CERN Controls Example

"There is a need to keep an index of all the controls entities and their parameters coming from different controls systems. Each controls entity has a name, description and location. For every entity there might be several parameters that are characterized by their name, description, unit, quantity code, data type and system they are sent from. This database will be accessed and exchange data with some of the existing databases related to the accelerators controls. It will ensure that every parameter name is unique among all existing controls systems."



CERN School of Computing

Information Requirements – CERN Controls Example

Samples of the data that has to be stored:

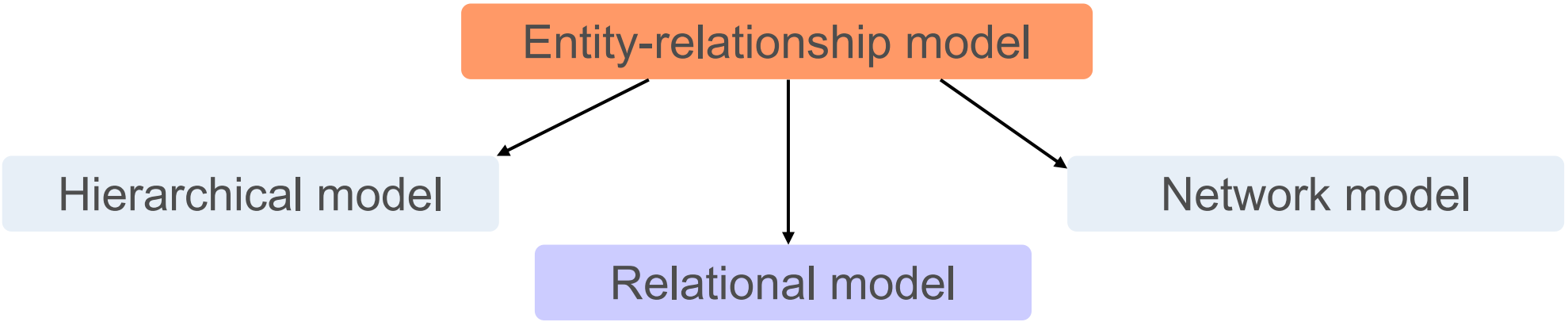
controls_entity
- name: VPIA.10020
- description: Vacuum Pump Sputter Ion type A in location 10020
- entity_code: VPIA
- expert_name: VPIA_10020
- accelerator: SPS
- location_name: 10020
- location_class: SPS_RING_POS
- location_class_description: SPS Ring position

entity_parameter
- name: VPIA.10020:PRESSURE
- description: Pressure of Vacuum Pump Sputter Ion type A in location 10020
- expert_name: VPIA.10020.PR
- unit_id: mb
- unit_description: millibar
- data_type: NUMERIC
- quantity_code: PRESSURE
- system_name: SPS_VACUUM
- system_description: SPS Vacuum

CERN School of Computing

Entity-Relationship Model

- The Entity-Relationship model (ER) is the most common conceptual model for database design nowadays
- No attention to efficiency or physical database design
- Describes data as *entities*, *attributes*, and *relationships*
- It is assumed that the Entity-Relationship diagram will be turned into one of the other available models during the logical design



CERN School of Computing

Entity

- A thing of significance about which the business needs to store information

  trivial example: employee, department
  CERN controls example: controls_entity, location, entity_parameter, system, quantity_code, data_type

- Entity instance – an individual occurrence of a given entity

  "a thing that exists and is distinguishable" J. Ullman

  trivial example: a single employee
  CERN controls example: a given system (e.g. SPS Vacuum)

Note: Be careful when establishing the 'boundaries' for the entity, e.g. entity employee – all employees in the company or all employees in a given department – depends on the requirements

CERN School of Computing

Attributes

- Attributes are properties which describe the entity
  attributes of system - name, description
- Attributes associate with each instance of an entity a value from a domain of values for that attribute
  set of integers, real numbers, character strings
- Attributes can be
  : optional
  : mandatory
- A Key - an attribute or a set of attributes, whose values uniquely identify each instance of a given entity

SYSTEM
id
description

ER Modeling Conventions

- If you use Oracle Designer the following convention is used:

ENTITY

- Soft box
- Singular name
- Unique
- Uppercase

ENTITY_PARAMETER

id
* description
o expert_name
* unit_id
* unit_description

attribute

- Singular name
- Unique within the entity
- Lowercase
- Mandatory (*)
- Optional (o)
- Unique identifier (#)

Note: There are different conventions for representing the ER model!

CERN School of Computing

Relationships

- Associations between entities

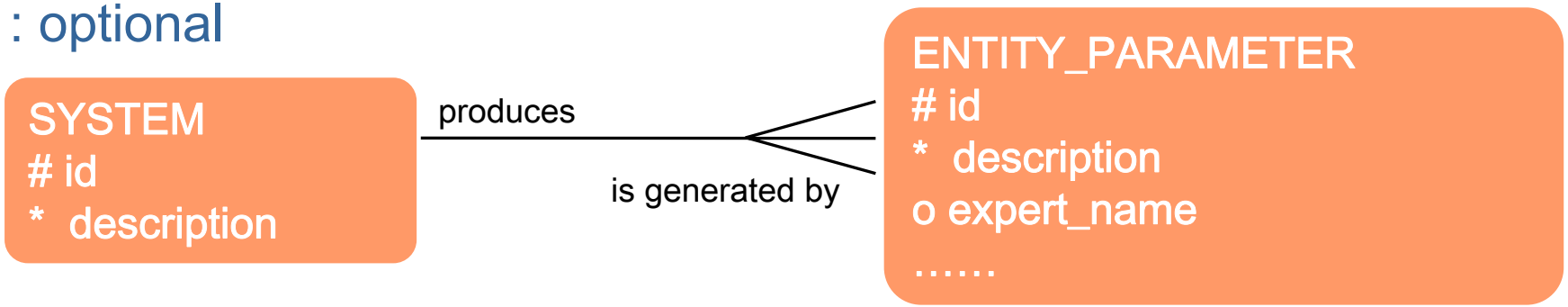
  examples: employees *are assigned to* departments
  entity_parameters *are generated by* systems

- Degree - number of entities associated with a relationship (most common case - binary)
- Cardinality - indicates the maximum possible number of entity occurrences
- Existence - indicates the minimum number of entity occurrences

set of integers, real numbers, character strings

: mandatory

: optional



CERN School of Computing

Relationship Cardinality

- One-to-One (1:1)

  one manager is a head of one department

Note: Usually this is an assumption about the real world that the database designer could choose to make or not to.

- One-to-Many (1:N)

  one system could generate many parameters
  one parameter is generated by only one system

- Many-to-Many (N:M)

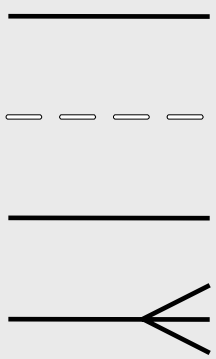
  many employees are assigned to one project
  one employee is assigned to many projects

ER Modeling Conventions

- If you use Oracle Designer the following convention is used:

Relationship

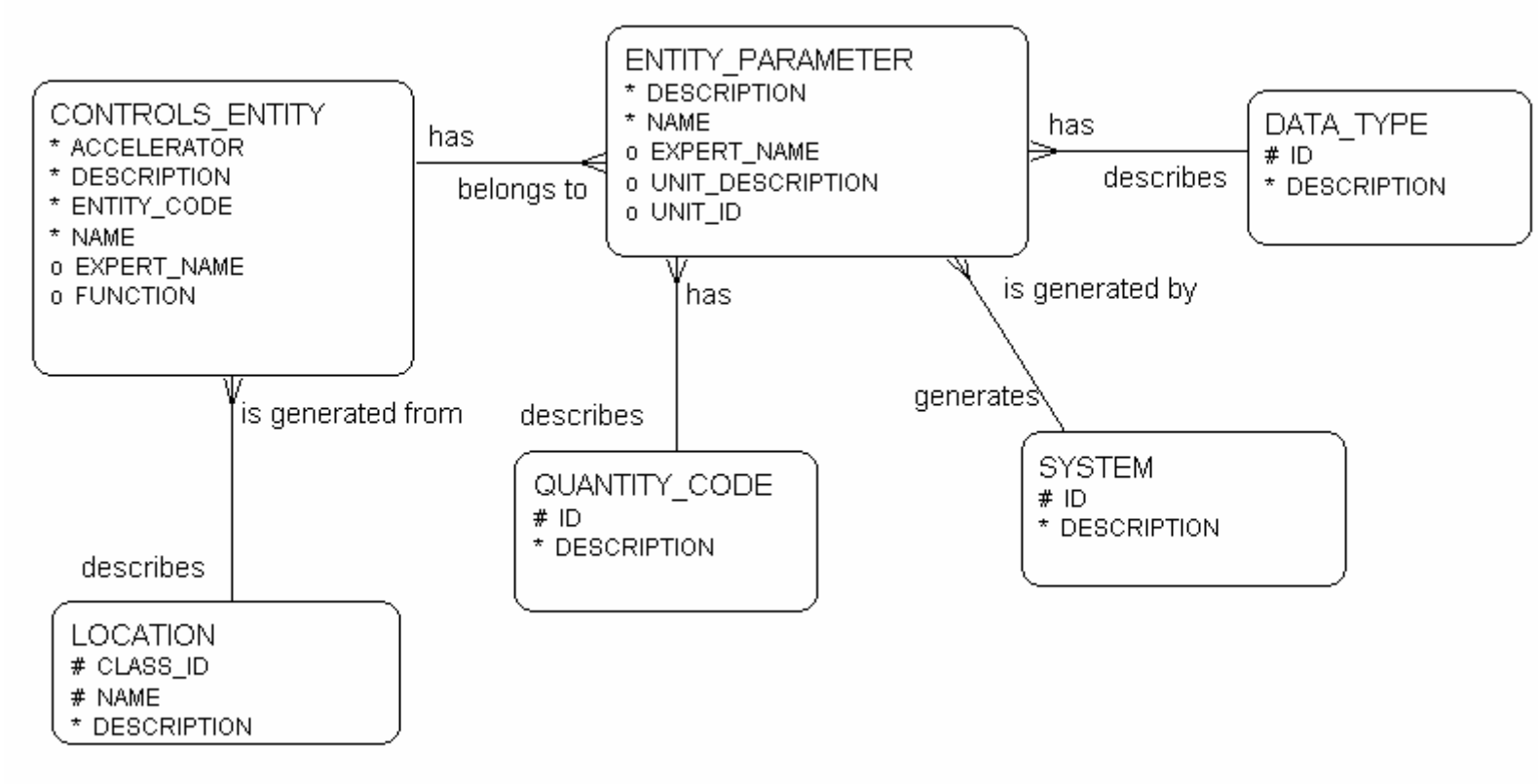
- Name – descriptive phrase
- Line connecting to entities
- Mandatory - solid line
- Optional - dashed line
- One - single line
- Many - crow's foot



Note: There are different conventions for representing the ER model!

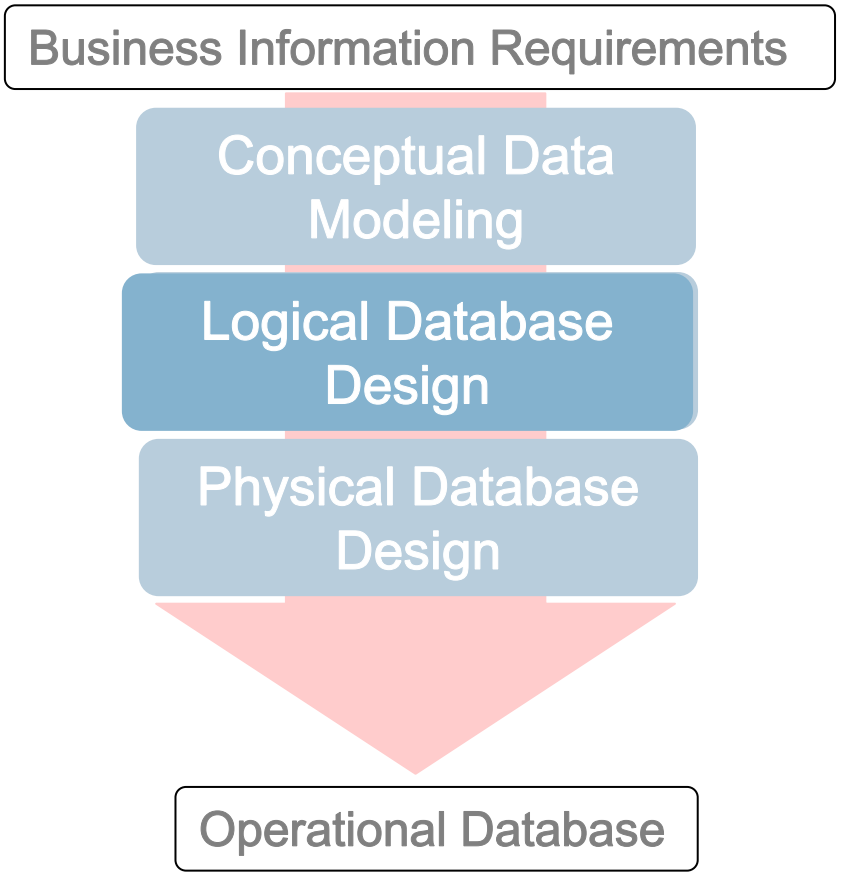
CERN Controls Example

- Entity-Relationship Diagram



CERN School of Computing

Logical Design



- Translate the conceptual representation into the logical data model supported by the DBMS

CERN School of Computing

Relational Model

- The most popular model for database implementation nowadays
- Supports powerful, yet simple and declarative languages with which operations on data are expressed
- Value-oriented model
- Represents data in the form of relations
- Data structures – relational tables
- Data integrity – tables have to satisfy integrity constraints
- Relational database – a collection of relations or two-dimensional tables

CERN School of Computing

Relational Table

- Composed by named columns and unnamed rows
- The rows represent occurrences of the entity
- Every table has a unique name
- Columns within a table have unique names
- Order of columns is irrelevant
- Every row is unique
- Order of rows is irrelevant
- Every field value is atomic (contains a single value)

CERN School of Computing

Primary Key (PK)

- A column or a set of columns that uniquely identify each row in a table
- Composite (compound) key
- Role – to enforce integrity
  : every table must have a primary key
- For every row the PK
  : must have a non-null value
  : the value must be unique
  : the value must not change or become ‘null’ during the table lifetime
- Columns with these characteristics are *candidate keys*

CERN School of Computing

Foreign Key (FK)

- Column(s) in a table that serves as a PK of another table
- Enforces referential integrity by completing an association between two tables

CERN School of Computing

Data Integrity

- Refers to the accuracy and consistency of the data by applying integrity constraints rules
- Attributes associate with each instance of an entity a value from a domain of values for that attribute

Constraint type	Explanation
Entity Integrity	No part of a PK can be NULL
Referential Integrity	A FK must match an existing PK value or else be NULL
Column Integrity	A column must contain only values consistent with the defined data format of the column
User-defined Integrity	The data stored in the database must comply with the business rules

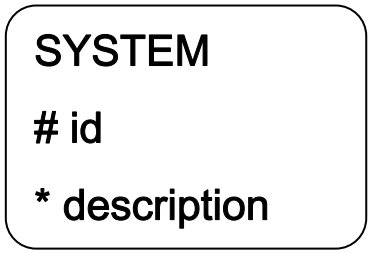
From Entity-Relationship Model to Relational Model

Entity-Relationship model

- Entity
- Attribute
- Key
- Relationship

Relational model

- Relational table
- Column (attribute)
- Primary Key (candidate keys)
- Foreign Key



SYSTEMS	
PK	SYS_ID
	SYS_DESCRIPTION

Relationships Transformations

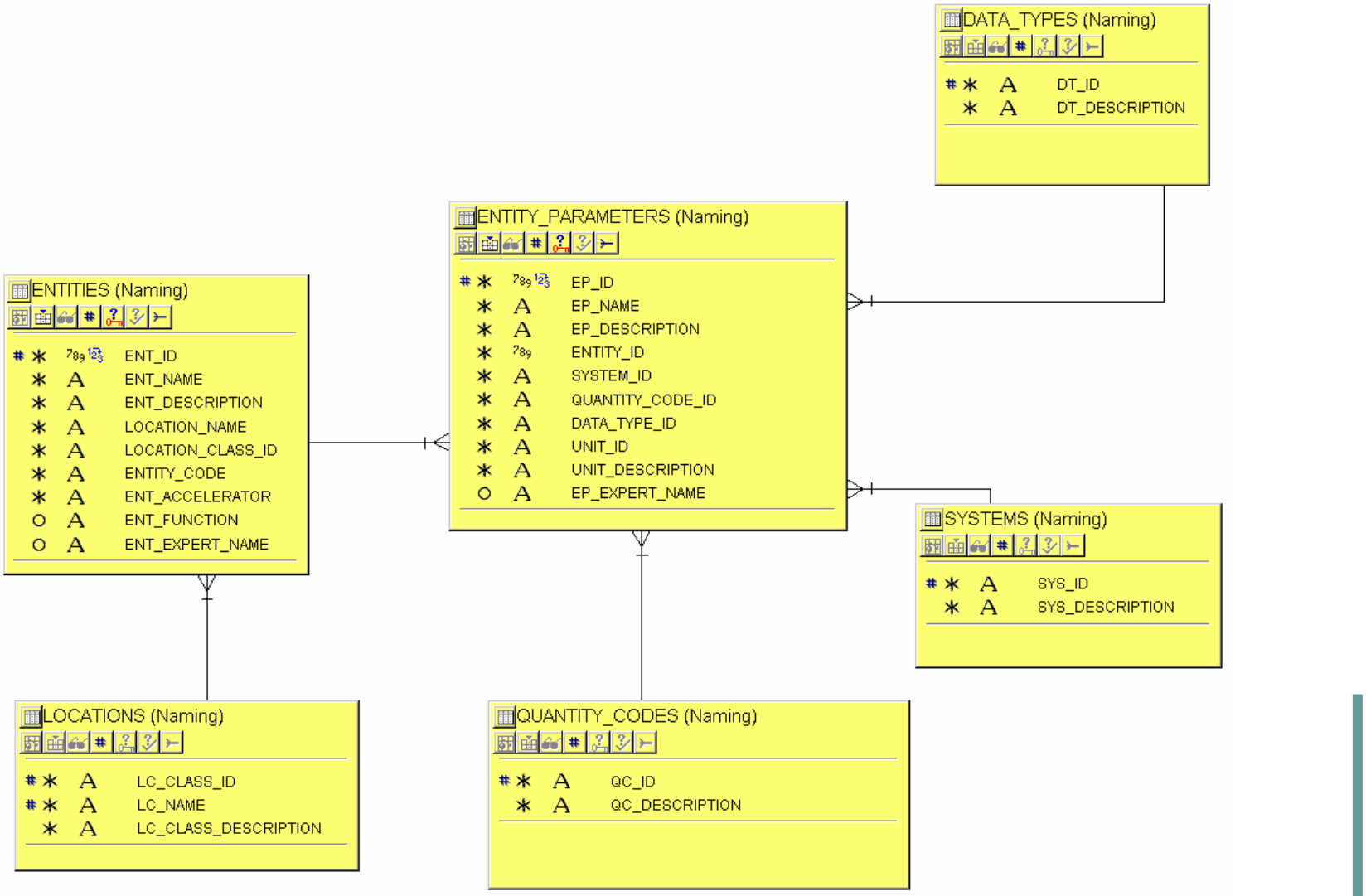
- Binary 1:1 relationships
  Solution : introduce a foreign key in the table on the optional side

- Binary 1:N relationship
  Solution : introduce a foreign key in the table on the 'many' side

- M:N relationships
  Solution : create a new table;
  : introduce as a composite Primary Key of the new table, the set of PKs of the original two tables

CERN Controls Example

- Relational Model – before normalization



Normalization

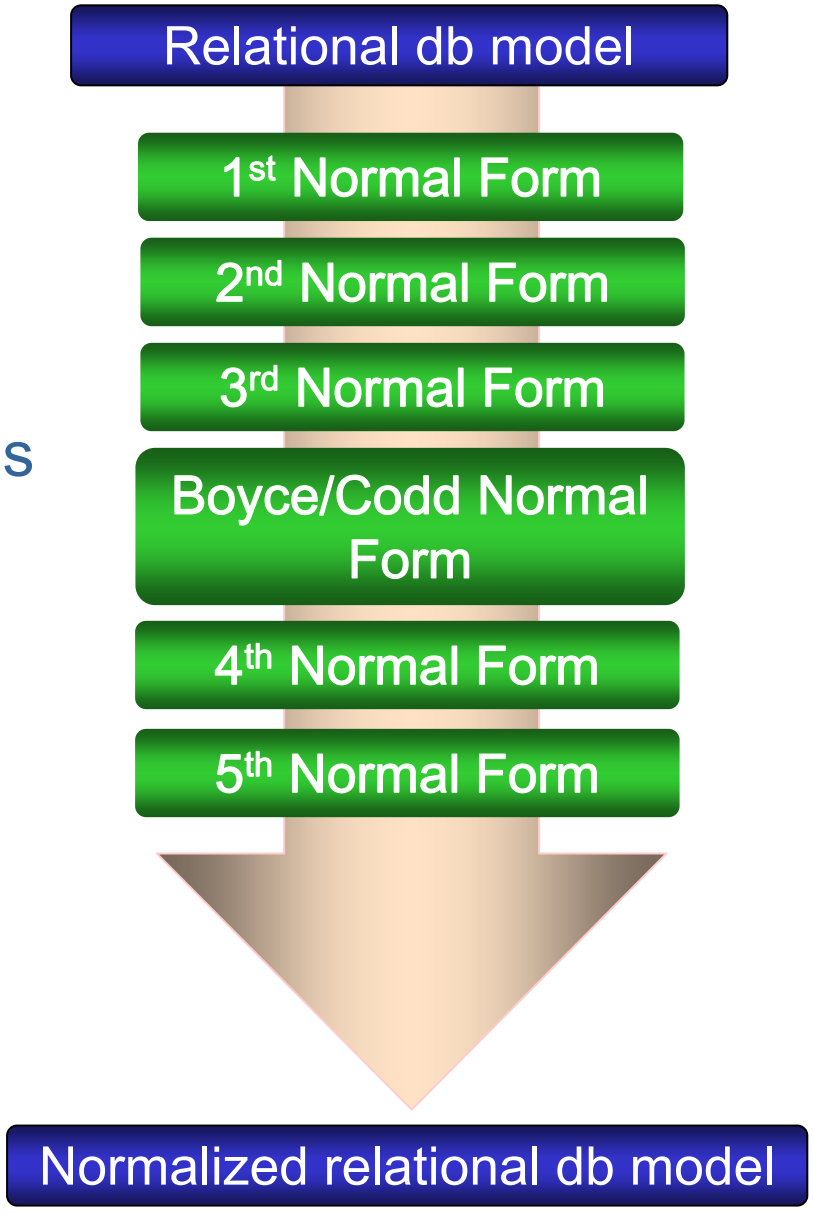
- A series of steps followed to obtain a database design that allows for consistent storage and avoiding duplication of data
- A process of decomposing relationships with ‘anomalies’
- The normalization process passes through fulfilling different Normal Forms
- A table is said to be in a certain normal form if it satisfies certain constraints
- Originally Dr. Codd defined 3 Normal Forms, later on several more were added

CERN School of Computing

Normalization

- Normalization process

- For most practical purposes databases are considered normalized if they adhere to 3rd Normal Form



1st Normal Form

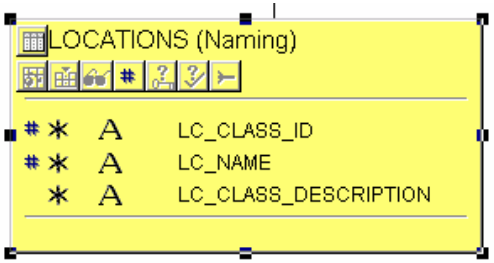
- 1st Normal Form - All table attributes' values must be atomic
  : multi-values are not allowed
- By definition a relational table is in 1st Normal Form

Definition: functional dependency (A -> B)
If attribute B is functionally dependent on attribute A, then for every instance of A you can determine the value of B

2nd Normal Form

- 2nd Normal Form - Every non-key attribute is fully functionally dependent on the PK
  : no partial dependencies
  : every attribute must be dependent on the entire PK

LOCATIONS(lc_class_id, lc_name, lc_class_description)



Solution:
: for each attribute in the PK that is involved in a partial dependency, create a new table
: all attributes that are partially dependent on that attribute should be moved to the new table

LOCATIONS (loc_class_id, loc_name)
LOCATION_CLASSES (lc_class_id, lc_class_description)

CERN School of Computing

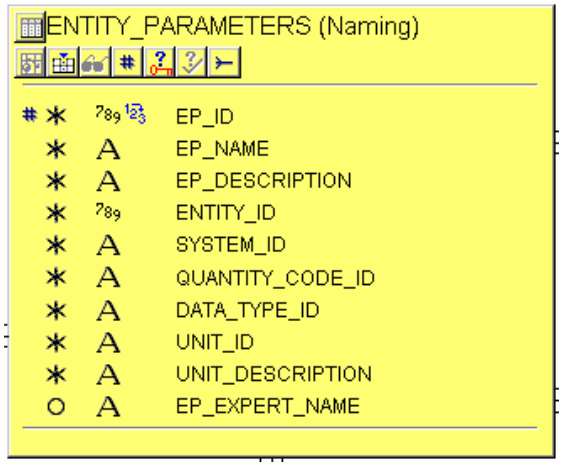
3nd Normal Form

- No *transitive dependencies* for non-key attributes

Definition: Transitive dependence

When a non-key attribute depends on another non-key attribute.

ENTITY_PARAMETERS(ep_id,…,unit_id, unit_description)



Solution:

: for each non-key attribute A that depends upon another non-key attribute B create a new table

: create PK of the new table as attribute B

: create a FK in the original table referencing the PK of the new table

ENTITY_PARAMETERS(ep_id,…,unit_id)
UNITS(unit_id, unit_descrption)

CERN School of Computing

Denormalization

- Queries against a fully normalized database often perform poorly

Explanation: Current RDBMSs implement the relational model poorly. A true relational DBMS would allow for a fully normalized database at the logical level, whilst providing physical storage of data that is tuned for high performance.

- Two approaches are used

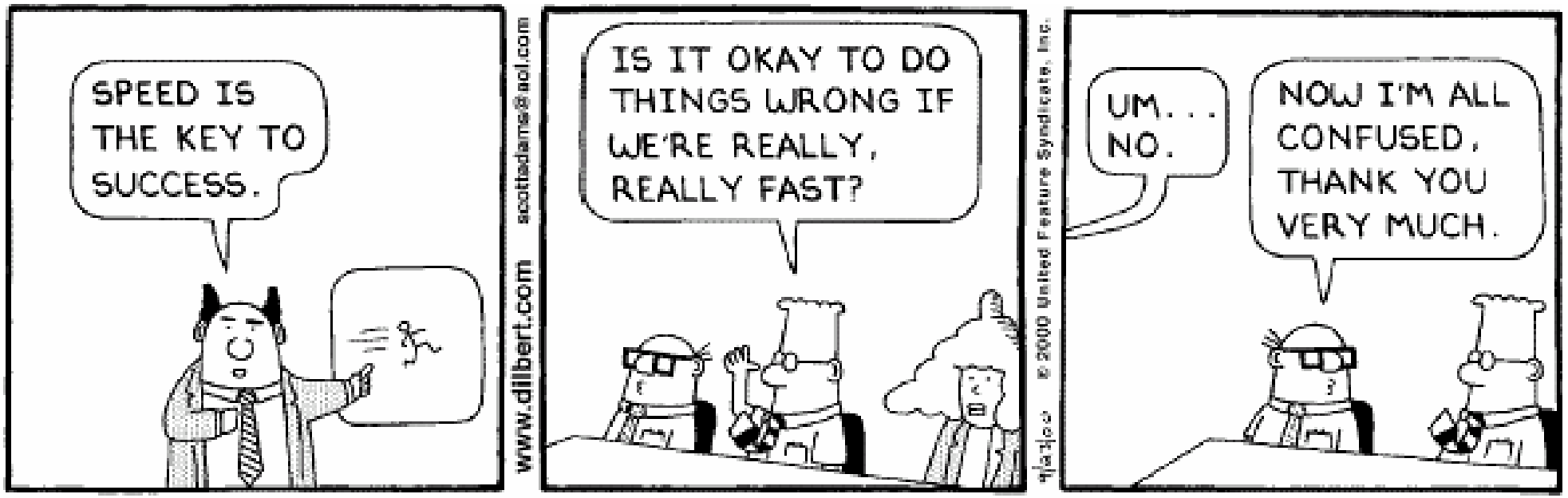
Approach 1: Keep the logical design normalized, but allow the DBMS to store additional redundant information on disk to optimize query response (indexed views, materialized views, etc.). In this case it is the DBMS software's responsibility to ensure that any redundant copies are kept consistent.

Denormalization

Approach 2: Use *denormalization* to improve performance, at the cost of reduced consistency

- Denormalization is the process of attempting to optimize the performance of a database by adding redundant data
- This *may achieve (may not!)* an improvement in query response, but at *a cost*
- There should be a *new set of constraints* added that specify how the redundant copies of information must be kept synchronized

Denormalization



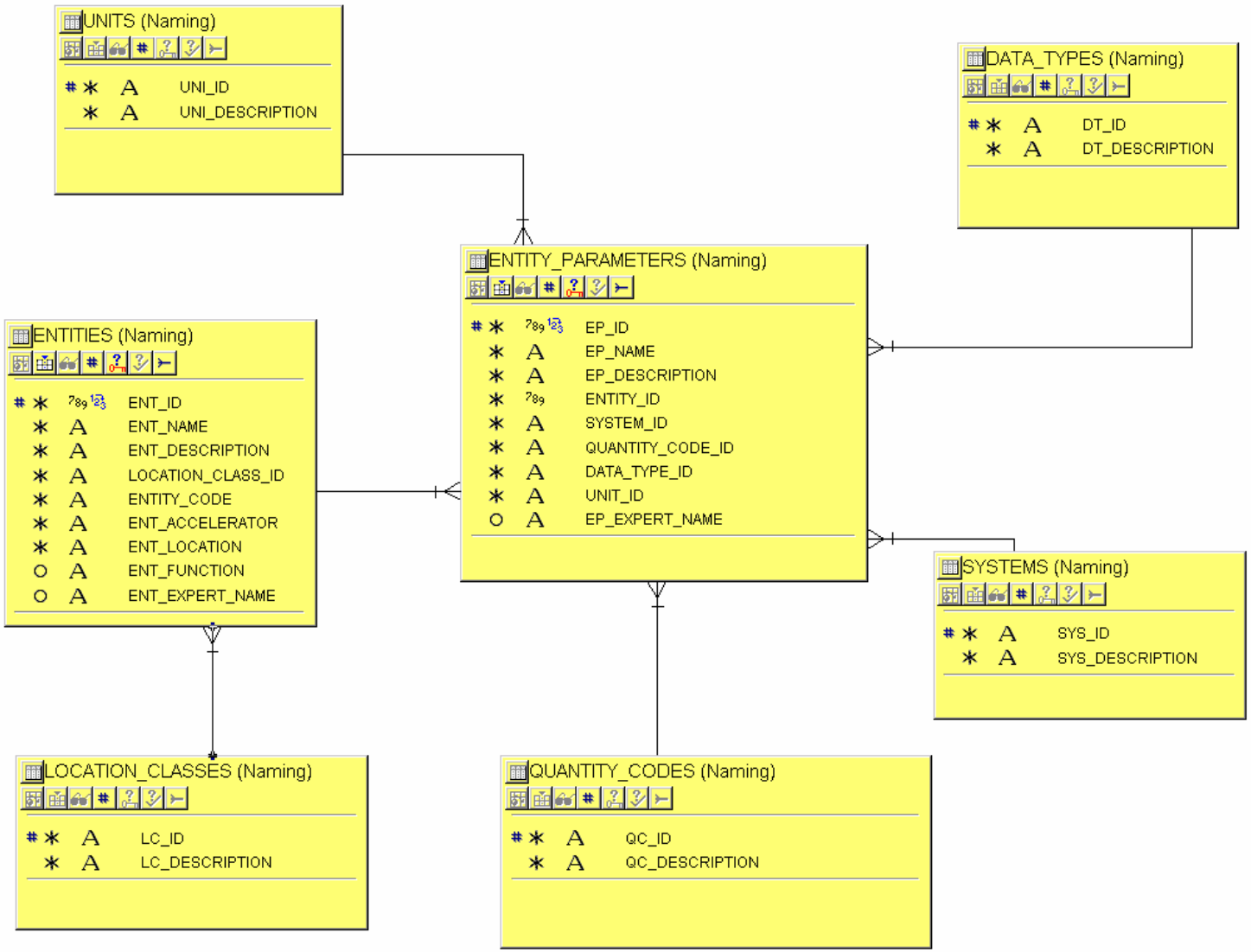
- Denormalization can be hazardous
  : increase in logical complexity of the database design
  : complexity of the additional constraints

- It is the database designer's responsibility to ensure that the denormalized database does not become *inconsistent*

CERN School of Computing

CERN Controls Example

- Relational Model – after normalization



CERN School of Computing

Structured Query Language

- Most commonly implemented relational query language
- SQL – originally developed by IBM
- Used to create, manipulate and maintain a relational database
- Official ANSI standard

CERN School of Computing

Structured Query Language

- Data Definition Language (DDL)
  : define the database schema
  : CREATE, DROP, ALTER table

- Data Manipulation Language (DML)
  : manipulate the data in the tables
  : SELECT, INSERT, UPDATE, DELETE

- Data Control Language (DCL)
  : control user access to the database schema
  : GRANT, REVOKE user privileges

CERN School of Computing

Database schema implementation

Definition: Database schema – a collection of logical structures of data

- The implementation of the database schema is realized through the DDL part of SQL
- Although there is a standard for SQL, there might be some features when writing the SQL scripts that are vendor specific
- Some commercially available RDBMS
  - : Oracle
  - : DB2 – IBM
  - : Microsoft SQL Server
  - : Microsoft Access
  - : mySQL

CERN School of Computing

Create Table

- Describe the layout of the table
  : table name
  : column names
  : *datatype* for each column
  : integrity constraints
    - column constraints, default values, not null
    - PK, FK

```
CREATE TABLE systems (
                        sys_id              VARCHAR2(20)
                       ,sys_description     VARCHAR2(100)
                      );
```

CERN School of Computing

Datatypes

- Each attribute of a relation (column in a table) in a RDBMS has a datatype that defines the domain of values this attribute can have
- The datatype for each column has to be specified when creating a table
- ANSI standard
- Oracle specific implementation

Oracle Datatypes

- CHAR (*size*) fixed-length char array
- VARCHAR2(*size*) variable-length char string
- NUMBER (*precision, scale*) any numeric
- DATE date and time with seconds precision
- TIMESTAMP data and time with nano-seconds precision
- CLOB char large object
- BLOB binary large object
- BINARY_FLOAT 32 bit floating point
- BINARY_DOUBLE 64 bit floating point
- *… + some others*

CERN School of Computing

Constraints

- Primary Key

```
ALTER TABLE systems
   ADD( CONSTRAINT SYSTEM_PK PRIMARY KEY (sys_id));
```

- Foreign Key

```
ALTER TABLE entity_parameters
   ADD (CONSTRAINT EP_SYS_FK FOREIGN KEY (system_id)
         REFERENCES systems(sys_id))
```

- Unique Key

```
ALTER TABLE entity_parameters
   ADD (CONSTRAINT EP_UNQ UNIQUE (ep_name));
```

Data Definition Language Statements

- Statements in the DDL

  : used for tables and other objects (views, sequences, etc.)

CREATE

ALTER

DROP

RENAME

TRUNCATE

```
CREATE SEQUENCE EP_SEQ
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   NOCACHE
```

CERN School of Computing

Best Practices in Database Design

- 'Black box' syndrome
  : understand the features of the database and use them

- Relational database or a data 'dump'
  : let the database enforce integrity
  : using the power of the relational database – manage integrity in multi-user environment
  : using PK and FK
  : not only one application will access the database
  : implementing constraints in the database, not in the client or in the middle tier, is faster
  : using the right datatypes

- Database independence

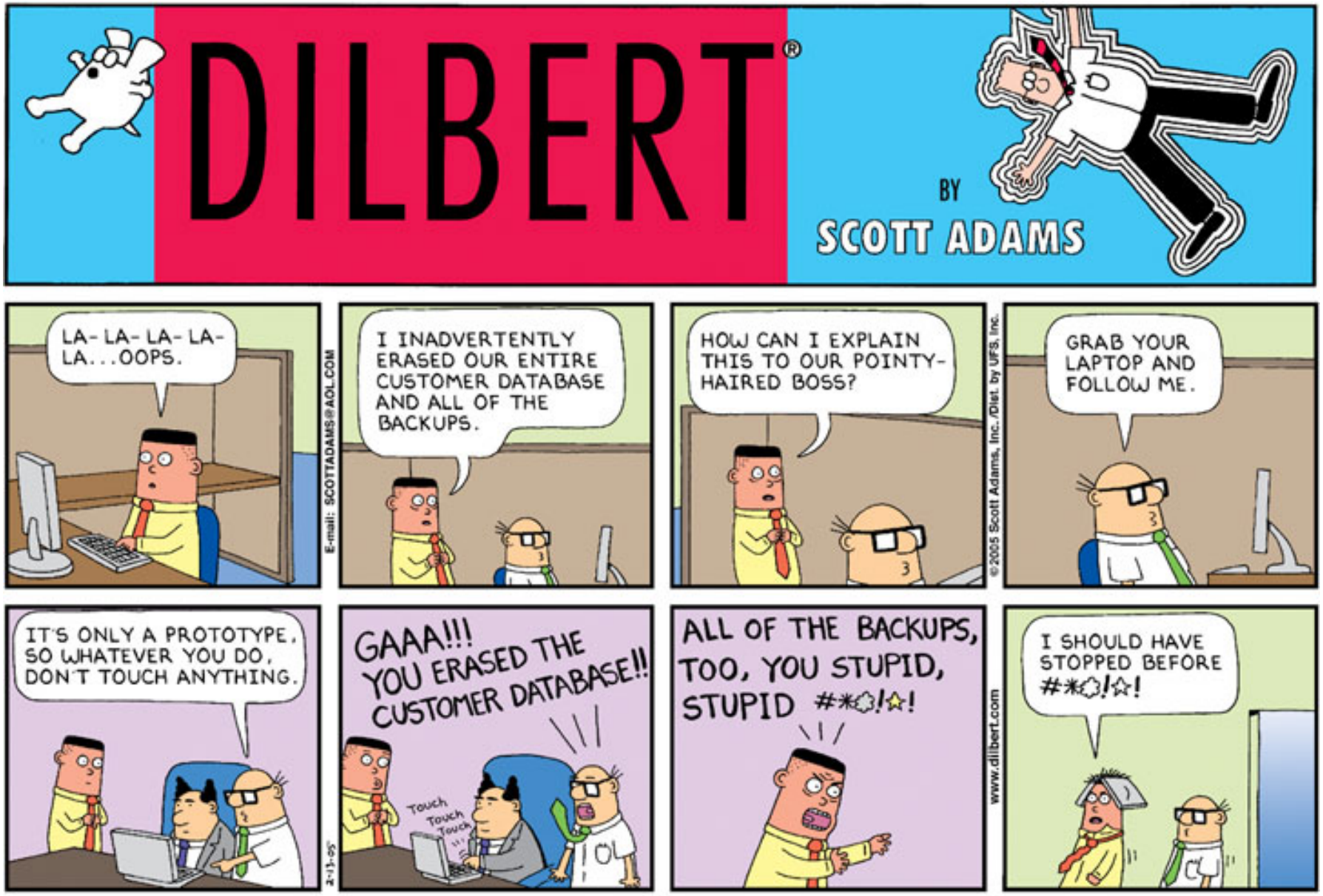
CERN School of Computing

Best Practices in Database Design

- Not using generic database models
  : tables - objects, attributes, object_attributes, links
  : performance problem!
- Designing to perform
- Creating a development (test) environment
- Testing with real data and under real conditions

CERN School of Computing

Best Practices in Database Design



CERN School of Computing

Development Tools

- Oracle provided tools
  : Oracle Designer
  : SQL* Plus
  : JDeveloper

- Benthic Software - http://www.benthicsoftware.com/
  : Golden
  : PL/Edit
  : GoldView
  : at CERN - G:\Applications\Benthic\Benthic_license_CERN.html

- Microsoft Visio

- CAST - http://www.castsoftware.com/
  : SQL Code-Builder

CERN School of Computing

References

[1] Ensor, D., Stevenson, I., *Oracle Design*, O'Reilly, 1997

[2] Kyte, T., *Effective Oracle by Design*

[3] Loney, K., Koch, G., *Oracle 9i – The Complete Reference*, McGraw-Hill, 2002

[4] Oracle course guide, *Data Modeling and Relational Database Design*, Oracle, 1996

[5] Rothwell, D., *Databases: An Introduction*, McGraw-Hill, 1993

[6] Ullman, J., *Principles of Databases and Knowledge-Base Systems volumn 1*, Computer Science Press, 1988

[7] Oracle on-line documentation
http://oracle-documentation.web.cern.ch/oracle-documentation/

End;

Thank you for your attention!

Zornitsa.Zaharieva@cern.ch