3. Schema Design:
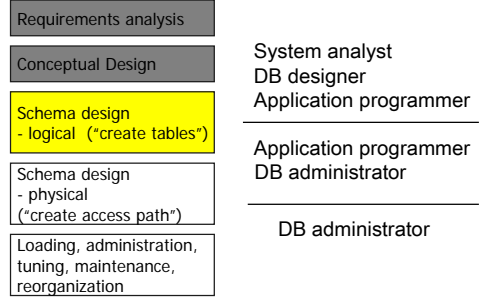### Logical Design using the Relational Data Model

3.1 Logical Schema Design
3.1.1 The Relational Data Model in a nutshell
3.1.2 Keys, candidate keys and more
3.2 From Conceptual to Logical Schema:
 Mapping ER to RDM
3.2.1 Relationships to tables
3.2.2 Consolidation
3.2.3 Mapping generalization hierarchies and more
3.2.4 Enforcing constraints
3.3 Triggers
3.4 Data types
3.5 Metadata Management

Kemper/ Eickler: chap. 3.1-3.3, Elmasri / Navathe: chap. 9
SQL/DDL: Melton/Simon: chap 2, 3.3, 4
 System documentation (e.g. Postgres, Oracle, MySQL, see references)

Context

Freie Universität Berlin

Requirements analysis	
Conceptual Design	System analyst
	DB designer
	Application programmer
Schema design - logical ("create tables")	
Schema design - physical ("create access path")	Application programmer
	DB administrator
Loading, administration, tuning, maintenance, reorganization	DB administrator

© HS-2010

04-DBS-ER-RDM-2

3.1 Logical schema design

Freie Universität Berlin

Logical Schema design is the transformation of the conceptual schema (e.g. ERM) into the logical schema (e.g RDM)

Easy: Algorithmic transformation
 using development tools
 (Oracle, Visio, DBDesigner, several Eclipse plugins,...)

Main concerns:
 - how to map relationships to tables
 - how to represent integrity constraints

© HS-2010

04-DBS-ER-RDM-3

3.1.1 The Relational Data Model in a nutshell

Freie Universität Berlin

The Relational Data Model
 Simplicity and formal rigor as the guiding principle
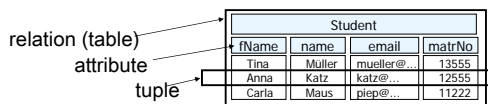
 KISS - Keep It simple, students

 Basically: an **algebra of tables**

 Table: data structure with a **fixed number of named columns** and an **arbitray number of rows** .

© HS-2010

04-DBS-ER-RDM-4

Basics

Freie Universität Berlin

relation (table)
attribute
tuple

Student			
fName	name	email	matrNo
Tina	Müller	mueller@...	13555
Anna	Katz	katz@...	12555
Carla	Maus	piep@...	11222

Relation schema (simplified notation omitting types):
```
Student(fname, name, email, matrNo)
```

SQL Data Definition Lanuage
```
CREATE TABLE Student(
   fname     VARCHAR (20),
   name      VARCHAR (30)   NOT NULL ,
   email     VARCHAR (40),
   matrNo    INTEGER )
```

© HS-2010

04-DBS-ER-RDM-5

Properties of the RDM

Freie Universität Berlin

- **No duplicate** rows   R is a set
- **No tuple order**
- Attributes have a **primitive type**, **no constructed type**   most DBS today allow constructed and multivalued types
- **single-valued**
- Attributes may have no value (**NULL** value)
- **Integrity constraints** must hold for all states of the DB over time ("invariant")
- **Unique names** in the relation and the DB namespace using dot-notation: R.a, db.S.b
- Database relations are **time variant** update, insertion, deletion of tuples

© HS-2010

04-DBS-ER-RDM-6

1

Keys and candidate keys

Freie Universität Berlin

Def.: A **key** of R(a1,…,an) is a subset of its
   its attributes, which **uniquely determines**
   the **tuples (= rows)** of R **and is minimal**

```
CREATE TABLE Student(
   fname        VARCHAR (20),
   name         VARCHAR (30)NOT NULL,
   email        VARCHAR (40),
   matrNo       INTEGER PRIMARY KEY)
```

But `matrNo ++ name` is not a PK – not minimal

But two or more attributes together *may* constitute a key:
   `name ++ fname` ??

© HS-2010   04-DBS-ER-RDM-7

Primary and Candidate keys

Freie Universität Berlin

(i)   A relation R may have **more than one potential key,**
      i.e. identifying, minimal attribute subset of R.
(ii) A potential key of R is called **Candidate key** * of R.
(iii) The **Primary Key** of R is an arbitrary candidate key

```
CREATE TABLE Employee(
   fname        VARCHAR (20),
   name         VARCHAR (30)NOT NULL,
   birthdate    DATE,
   email        VARCHAR (40),
   jobDesc      VARCHAR(200)
    CONSTRAINT region_pk
    PRIMARY KEY (name,birthdate))
```

Primary key with
more than one
attribute as
separate, **table
constraint**.

user defined **constraint name**, .. why?

© HS-2010   * Schlüsselkandidat – nie Kandidatenschlüssel   04-DBS-ER-RDM-8

Artificial Keys

Freie Universität Berlin

Sometimes useful, to assign an artificial key to relation R

```
CREATE TABLE Employee(
   p#           INTEGER PRIMARY KEY,
   fname        VARCHAR (20),
   name         VARCHAR (30),
   ...)
```

Not an artificial key,
exists in reality.

```
CREATE TABLE LogRecords(
   seq#         INTEGER PRIMARY KEY,
   logType      CHAR,
   logEntry     VARCHAR (300),
   time         TIMESTAMP)
```

Artificial, should be
e.g.: 1,2....
Postgres: **SERIAL**
Oracle:use
sequence generator

Surrogates: system internal row keys for special purposes

© HS-2010   04-DBS-ER-RDM-9

Operations on tables

Freie Universität Berlin

Why "**Relational Algebra**"?

Employee			
fName	name	deptm	p#
Tina	Müller	IT	13555
Anna	Katz	Sales	12555
Carla	Maus	IT	11222

Department		
name	boss	location
Sales	11234	B
Acc	12222	MUE
IT	13555	B

Operations on tables result in tables!
e.g.   select some rows: "Employees in IT-Department"
        project columns: "Names of all employees"
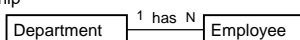   "join" columns: "Department location of Tina Müller"

Employee			
fName	name	deptm	p#
Tina	Müller	IT	13555
Anna	Katz	Sales	12555
Carla	Maus	IT	11222

Department		
name	boss	location
Sales	11234	B
Acc	12222	MUE
IT	13555	B

© HS-2010   04-DBS-ER-RDM-10

RDM: Foreign Keys

Freie Universität Berlin

```
CREATE TABLE Employee(
   p#           INTEGER PRIMARY KEY,
   fname        VARCHAR (20),
   name         VARCHAR (30),
   deptm        VARCHAR(20)
        FOREIGN KEY FK_Dep REFERENCES Department

CREATE TABLE Department(
   name         VARCHAR (20) PRIMARY KEY,
   boss         Integer,
   location     VARCHAR (5)
        Foreign Key FK_Boss REFERENCES Employee )
```

Implements 1:N relationship

Department —1 has N— Employee

© HS-2010   04-DBS-ER-RDM-11

Foreign key

Freie Universität Berlin

Def: A **foreign key** is one or more attributes FK of a
   relation S, with the properties:

   (1) attributes of FK have the **same domains** as the
        attributes of key* p_k of a **relation R**
   and
   (2) a **value of FK** in row of S **either occurs** as a
        value of the primary key for some row in **R**
        or is NULL.

Def.: Referential integrity of a database is preserved, if
   all (explicit) **foreign key constraint hold.**

   * Usually the primary key, but not required!

© HS-2010   04-DBS-ER-RDM-12

2

What next: From entities to tables

Freie Universität Berlin

Map E-R design to relational schema

Define relational schema, table names, attributes and types, invariants

Design steps:
- Translate **entities into relations**
- Translate **relationships** into relations
- **Simplify** (consolidate) the design
- Formal analysis of the schema   (postponed)
- Define **tables in SQL**
- Define additional **invariants**

3.2 From Conceptual to Logical schema…

Freie Universität Berlin

Entity (types) with keys $\Rightarrow$ tables (schema relations)
- Key attributes in the Conceptual model are primary keys in the RDM

Weak entities:

add primary key of superior entity to partial key of weak entity

Example: `country(c_id, name, gnp, ..)`
          `region(name, population, area,..)`

partial key

$\Rightarrow$ `region (c_id name,...)`

never NULL, why?

Mapping Relationships

Freie Universität Berlin

Relationships in general: tables (schema relations)

- Attributes: **keys of the involved relations**
                 and attributes of the relationship
- Key of the "relationship" table:
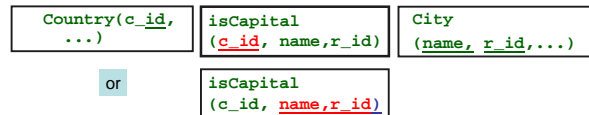 one or all keys of the related relations

Dep	(1,1) has	Empl

Has	d_name	emp#
	1	32
	2	47
	1	33

Each Employee has unique ёmp#
$\Rightarrow$ row in D_E identifies by emp# $\Rightarrow$ key is emp#

3.2.1 Relationships to tables

Freie Universität Berlin

1:1-relationship
Chose as key one of the keys of the involved relations

Country	(0,1) (1,1)	City
	isCapital	

Country(c_id, ...)	isCapital (c_id, name,r_id)	City (name, r_id,...)

or   `isCapital (c_id, name,r_id)`

Relationships to tables

Freie Universität Berlin

1:N relationship

	1 N	
Department	(1,1) (0,*)	Employee
	has	

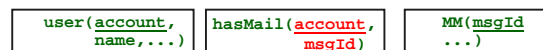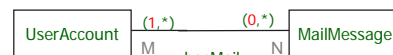`Dep( name, boss, … location)`   `has (p#, name)`   `Employee(p#, fname, ..)`

A table R representing an 1:N – E-R-relationship has
- **as attributes** the keys of both relations and relationship attribute – of any
- **as** its key the **key of the "N-side entity type"**

N:M relationship

Freie Universität Berlin

UserAccount	(1,*) (0,*)	MailMessage
	M hasMail N	

`user(account, name,...)`   `hasMail(account, msgId)`   `MM(msgId ...)`
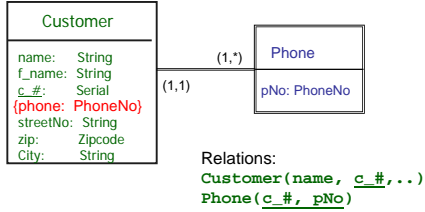
Neither **account** nor **msgId** alone have key property

A separate table R representing an M:N – E-R relationship has
- **as attributes** the keys of both relations and relationship attribute – of any
- as key the **keys of both entities**
  N-ary relationship: **all keys** make up the new key

3

Multi-valued attributes

Multiple value attribute
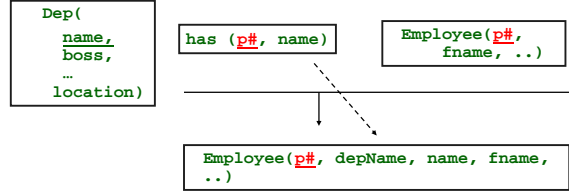⇨ weak entity with a single attribute

Customer

name:	String
f_name:	String
c_#:	Serial
{phone:	PhoneNo}
streetNo:	String
zip:	Zipcode
City:	String

(1,*) — **Phone**

(1,1) — pNo: PhoneNo

Relations:
```
Customer(name, c_#,..)
Phone(c_#, pNo)
```

... or array-type / list type for attribute (Oracle, PostgresSQL and others).

© HS-2010   04-DBS-ER-RDM-19

3.2.3 Consolidation

Def.: Consolidation (simplification) of a relational scheme is
the process of merging those table (schemas) having the *same key*
attributes **into one** table schema (recursively)

R(**k1,...,kn**, a1,...an), S(**k1,...,kn**, b1,...bm)
⇨ RS(**k1,...,kn**, a1,...an,b1,...,bm)

```
Dep(
    name,
    boss,
    …
    location)
```

```
has (p#, name)
```

```
Employee(p#,
    fname, ..)
```

```
Employee(p#, depName, name, fname,
    ..)
```

© HS-2010   04-DBS-ER-RDM-20

Example

1 : 1 - relationship

```
Country
(c_id, name,  ...)
```

```
isCapital
  (c_id,  name, region)
```

renamed,
path ex-
pression
Region.name
not allowed

⇨ `Country(c_id, name, capital, region,…)`

Foreign key

© HS-2010   04-DBS-ER-RDM-21

Consolidation: example

```
CREATE TABLE Country
(name            VARCHAR(32) NOT NULL,
 C_ID            VARCHAR(4)  PRIMARY KEY,
 population       INT,
 growth          NUMERIC(4,1),
 area                INT,
 GNP             INT,
 capital          VARCHAR(25) NOT NULL, -- renamed
 region          VARCHAR(4) NO NULL,
 type_of_Gov     VARCHAR(35),
 head_of_Gov     VARCHAR(70)
 CONSTRAINT fk_capital FOREIGN KEY
    (capital, region) REFERENCES City)
```

© HS-2010   04-DBS-ER-RDM-22
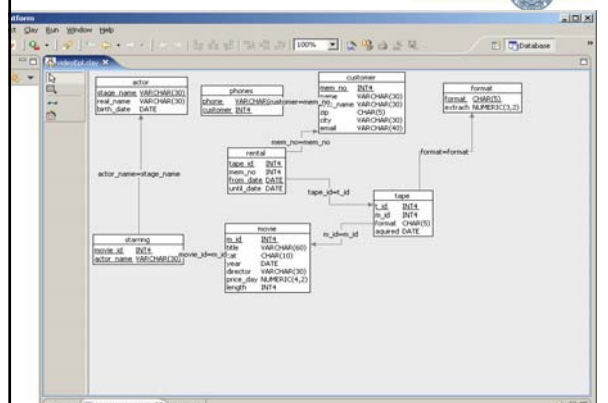

E-R to RDM mapping: discussion

Transformation

- unambiguous for relations representing 1:N
  relationship, ... but *consolidation optional*.
- 1:1 relationships: choice - merge with one of
  the "entity- tables"
- **M:N** relationships: <u>never merge</u>
  Represented **always by <u>separate tables</u>** in
  the RDM

Very simple process:
Many DB-Design tools model relationships
directly by means of foreign keys!

© HS-2010   04-DBS-ER-RDM-23

Schema reengineering

4

Discussion continued

Always merge 1:N relationships?

Example:
```
Person(id, name, phone#,...)
Room (rNo, building#, size, netSocket# ...)
Sits_in(id, rNo,b#, since, gotKey,
        numberOfKey, ...)
```
Merge would result in a relation with many NULL values

- **Merging 1:N relationships makes sense in most cases**
  - If relationship has many attributes do not merge when many NULL values expected
  - If attributes of relationship are used infrequently by applications, do not merge (*)

(*) efficiency argument: avoid unnecessary data transfers

Discussion(2)

Never merge M:N relationships

```
Person(id, name,...)
Hobby (hobby, kind, class_of_risk)
has_H(id, hobby, casualty)
```

⇨ Person(id, name,..., hobby, casualty...)
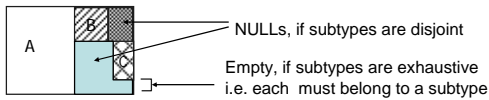
Key has been changed, redundancy introduced

3.2.3 E-R to RDM mapping: Generalization

First alternative: **One "big" A-table** with attributes from all specializations

$$A(\underline{x}, y, s, t)$$

NULLs, if subtypes are disjoint

Empty, if subtypes are exhaustive
i.e. each must belong to a subtype
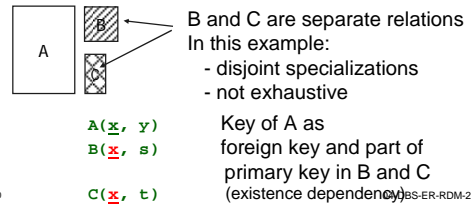

Generalization: separate tables

Second Alternative:

- **separate relations for A, B and C**
- make a one-to-one correspondence between every tuple from B and the appropriate A's
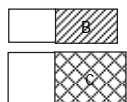- ..and the same for the C's

B and C are separate relations
In this example:
- disjoint specializations
- not exhaustive

```
A(x, y)
B(x, s)
C(x, t)
```

Key of A as foreign key and part of primary key in B and C
(existence dependency)

E-R to RDM mapping: Generalization

Third alternative
Extend A by B and C, respectively


```
AB(x,y,s)
AC(x,y,t)
```

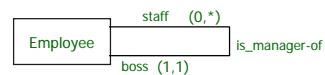Required: Subtypes must be exhaustive, i.e. complete specialization

Think about the **pros and cons** of each solution!

E-R to RDM mapping

Recursive relationships


```
Employee (eid, ….., managed_by_eid,…)
```

Transformation step depending on cardinalities
just like non-recursive relationships