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

3.1 Logical Schema Design

3.1.1 The Relational Data Model – Basics

3.1.2 Keys, candidate keys and more

3.2 From Conceptual to Logical Schema: Mapping ER to RDM

3.2.1 Relationships to relations: a simple step

3.2.2 Mapping weak entities and multivalued attributes

3.2.3 Consolidation

3.2.4 Mapping generalization hierachies and more

Kemper/ Eickler: chap. 3.1-3.3, Elmasri / Navathe: chap. 9

Context

- Requirements analysis
- Conceptual Design
- Schema design - logical ("create tables")
- Schema design - physical ("create access path")
- Loading, administration, tuning, maintenance, reorganization

System analyst
DB designer
Application programmer

Application programmer
DB administrator

DB administrator

3.1 Logical schema design

- Second phase of DB design
  - Transform the E-R model into the logical schema of a particular data model
  - Easy for Relational Data Model (RDM)
    can be performed automatically
    (e.g. Oracle Designer, Visio, and many other tools)
- The next steps:
  - Relational Data Model: Basics
  - General principles for mapping entities and relationships to relations (of the RDM)
  - Define the DB schema by means of a Data Definition Language (DDL) - SQL DDL in Relational DBS
  - Formal analysis of the relational schema

3.1.1 The Relational Data Model - Basics

The Relational Data Model

- Simplicity and formal rigor as the guiding principle

> KISS - Keep It simple, stupid
>
> Do we need entities and relations?
>
> No, just relations and attributes

- Relations are mathematical objects
- Relations may be implemented by tables
- Introduced by E.F. Codd, 1970, at that time at IBM Research labs, San Jose. Honored by the Turing award for his achievement.

3.1.1 Basics of the RDM

Basic ideas

Important terms

- Database is collection of relations
- Relation R = set of n-tuples
- Relation schema $R(a_1, a_2, \ldots a_n)$
- Attributes a_i have atomic values from domain $D_i = dom(a_i)$

relation (table)

attribute

tuple

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

relation schema:

```
Student(fname, name, email, matrNo)
```
or
```
Student(fname:Name, name:Name, email:EmailType,
                                  matrNo:number)
```


RDM

- Notation

  Relation $R \subseteq dom(a_1) \times dom(a_2) \times \ldots \times dom(a_n)$

  Attribute set $A = A(R) = \{a_1, a_2, \ldots, a_n\}$

  Tuple = $r \in R$

  Degree of R: number of attributes

  Relation Schema $R(a_1, a_2, \ldots, a_n)$

- Terminology

  different notations in use

Relation = table (file) — not the same: relations are sets. tables may have duplicate entries

Tuple = row, record

Attribute = field (component)

table, row, record, field mainly used in the context of an RDBS implementation

Properties of the RDM

Properties

- No duplicate rows: R is a set
- No tuple order
- Database relations are time variant
  update, insertion, deletion of tuples
- Integrity constraints must hold for all states over time
- Attributes have a primitive type, no constructed type
- Unique names in the relation and the DB namespace
  otherwise dot-notation: R.a, db.S.b
- Attributes single-valued (more or less…)
- Attributes may have no value (NULL value)

3.1.2 Keys, candidate keys and more

Key

Important term

- Each tuple identifiable by values of particular attributes ("key attributes")
- Key of R(a1,...,an): a minimal sequence of attributes which identify tuples

Example:
(first_Name, last_Name, birthdate, phone)
is a sequence of attributes which identify tuples
of relation "Student"
It is not minimal: (first_Name,last_Name, birthdate)
will do

RDM: key

Formally: $(k_1,\dots,k_k) \subseteq A = \{a1,\dots,an\}$ is a key iff:

1) if $v_1,\dots,v_k$ are the values of $k1,\dots k_k$ of a row r1 and $v_1',\dots,vk'$ for a row r2, and r1 != r2
   $\Rightarrow$ there is at least one i, 1 <= i <= k, $v_i != v_i'$)
   (identifying property)

2) there is no subset of $\{k_1,\dots,k_k\}$ with this property
   (minimality)

each relation R has a key since R is a set

(in theory ...)

Keys and Superkeys

– Frequently: a single artificial attribute, e.g. a number
– Similar to oid in OO systems
– Does not always make sense! (e.g. student administration system!)

Super key

A super set SK of a key of R(a1,...,an) with attributes
$\{k_1,\dots,k_k\} \subseteq SK \subseteq \{a1,\dots,an\}$ is called a super key of R

- Example `Student(fname, name, email, matrNo, major, birthdate)`
  Key: `matrNo`, Super key: {`matrNo, name`}

Primary Key, candidate key

Important terms

- A relation must have a key, but may have several.
- The keys of a table are called candidate keys*

Example:
Student(fname, name, email, matrNo, major)
Candidate Keys: **(Email), (MatrNo)**

- Primary Key: an arbitrarily chosen candidate key
- Primary key access is "by value", not by location (oid) as in OO-languages

Example:

```
Find Student.name where email='katz@inf.fu-berlin.de'
```

* Schlüsselkandidaten, nicht Kandidatenschlüssel

RDM: Foreign Key

Important term

How can rows of other tables be referenced?

Example:

R: Student(fname, name, email, matrNo)
S: Exam (prof, std, subject, grade, dateTime)

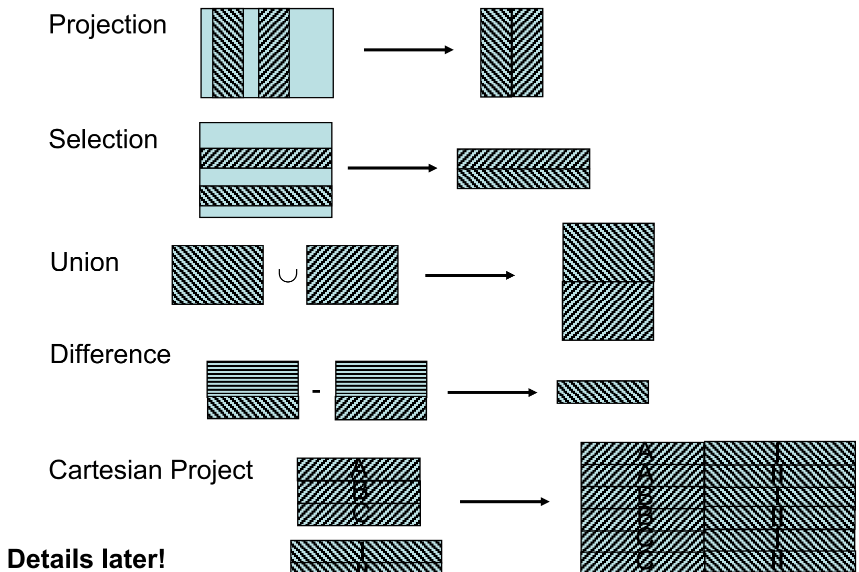
Exam.std should identify exactly one student.

Foreign key

A set of attributes FK in relation schema S is called a foreign key if

- attributes of FK have the same domain as the attributes of primary key pk of a relation R
- A value of FK in tuple t_s of S either occurs as a value of pk for some t_r in R or is NULL.

RDM: basic operations schematically



What next: From Entities to relations

1. Select data model
   → here relational data model
2. Transform conceptual model into logical schema of relational data model

Mapping E-R designs to relational schemas

- Define relational schema, table names, attributes and types, invariants
- Design steps:
  1. Translate entities into relations
  2. Translate relationships into relations
  3. Simplify the design
  4. (Select database system)
  5. Define tables in SQL
  6. Define additional invariants
  7. Formal analysis of the schema

3.2 From Conceptual to logical schema…

Mapping to relational schemas – DB System independent

- Each entity is a relation (entity type -> relation scheme)
  - Key attributes in the Conceptual model are keys in the RDM
- Each relationship is a relation (of the RDM)
  - Attributes: keys of the involved relations and attributes of the relationship, if any
  - Key: is composed of some or all of the keys of the related instances

Examples:

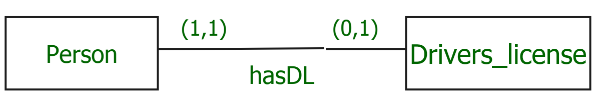
Keys are underlined

```
Tape(id, acqDate)
Movie(id, title, category, price_day,
      director, year) // synthetic key introduced
recorded_on(tapeId, movieId)   // renaming
```

3.2.1 Relationships to relations

1:1-relationship

Chose as key one of the keys of the involved relations




```
Person(id,
  ...)
```

```
hasDL (persId,
          license#)
```

```
DL(license#,
  ...)
```

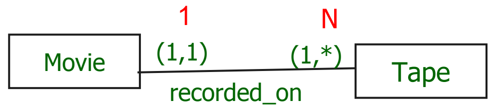
or

```
hasDL (perId,
          license#)
```

Both **perID** and **license#** each have the key property

Relationships to relations

1:N relationship




```
Movie(
  id,
  title,
  ….
  year)
```

```
Rec_on (tapeId,
        movieId)
```

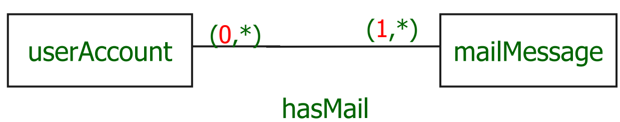
```
Tape(id,
     acqDate)
```

A separate relation R representing an 1:N – E-R-relationship has as its key the key of the "N-side relation" or in (min,max)-Notation the (0,*) or (1,*) side.

or mathematically:
If E1 -> E2 is a function, then the key of the relation corresponding to E1 is the key of R

Logical Design E-R to RDM mapping

N:M relationship




```
user(account,
     name,...)
```

```
hasMail(account,
        msgId)
```

```
MM(msgId
   ...)
```

- keys of both together form the key of R or part of the key in case of attributed relationships
- Neither **account** nor **msgId** alone have key property

E-R to RDM mapping: first step

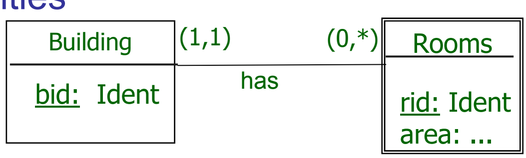
- Case study after simple mapping

```
Tape(id, acqDate)
Movie(id, title, category, price_day,
        director, year, length)
Customer(mem_no,name,f_name, zip,
          city, streetNo)
Actor(stageName, real_name, birth_date)
Format(format, extraCh)
Rec_on (tapeId, movieId)   // renaming
 Has  (tapeId, format)      // renaming
Starring (stageName, movieId)
```

Rental ?? Weak entity ?!, **phoneNo**: MV attrib.

Logical Design E-R to RDM mapping

Weak entities



- For each weak entity WE create relation R
- Include all attributes of WE
- Add key of identifying entity to weak entities (relative) key
- Part of key is a foreign key

Relational Schema:

```
Building(bid)
Rooms(cid, bid, area,...)
```

Note: weak entities have to be transformed before relationships are transformed

Logical Design E-R to RDM mapping

- Example

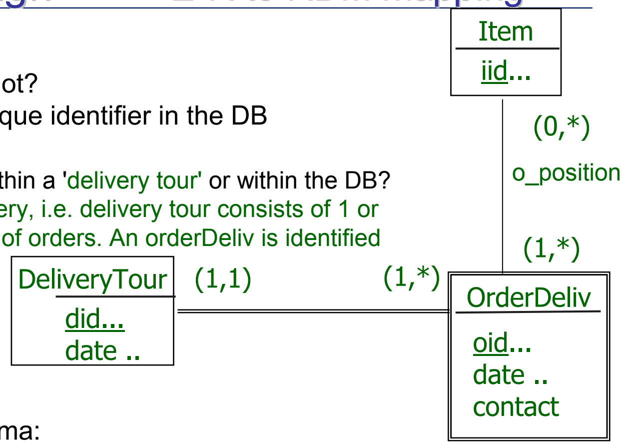
Weak entity or not?

Item: has a unique identifier in the DB

Order: ?

Is oid unique within a 'delivery tour' or within the DB?

Assumed: delivery, i.e. delivery tour consists of 1 or more deliveries of orders. An orderDeliv is identified within a tour.

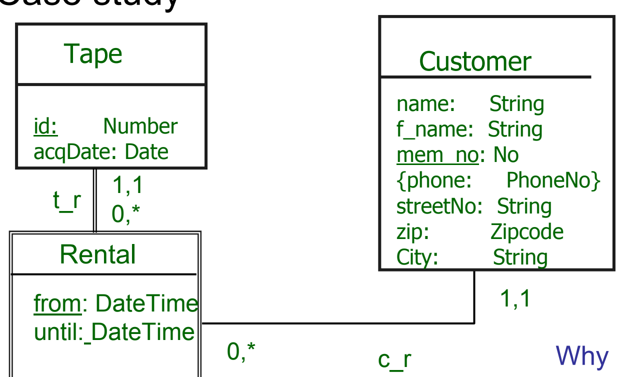


Relational Schema:

```
DeliveryTour(did, date)
OrderDeliv(oid, did, date, contact)
O_Position(iid, oid, did)
Item(iid, ... )
```


Weak entities: case study

Case study



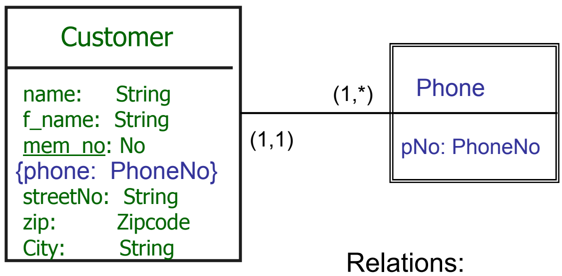

```
Tape ( id, ...)
Customer(name, mem_no,..)
Rental(id, from, until)
c_r (id, from, mem_no)
```

Why is "Rental" not weak entity dependent on customer?

...application semantics: customer lends more than one tape with the same 'from'-time stamp.

E-R to RDM mapping: Multiple values

Multiple value attribute ⇨ weak entity with a single attribute



Relations:

```
Customer(name, mem_no,..)
Phone(mem_no, pNo)
```

... or array-type for attribute (PostgresSQL and others).

3.2.3 E-R to RDM mapping: Consolidation

Simplification (consolidation)

of the relational DB schemas

Rule:

Merge those relation schemas with the *same key* attributes into one relation schema

$R(\underline{k_1,\ldots,k_n}, a_1,\ldots a_n)$, $S(\underline{k_1,\ldots,k_n}, b_1,\ldots b_m)$

⇨

$RS(\underline{k_1,\ldots,k_n}, a_1,\ldots a_n, b_1,\ldots,b_m)$

Remark: Attribute semantics must match, not the literal name

```
Tape    (id,aqDate)
Has     (id,format)

=> Tape(id, aqDate,format)
```

```
Tape(id, ....)
Recorded_on (tapeId, movieId)
=>   Tape(id, aqDate,format
          movieId)
```

E-R to RDM mapping: Simplification

Transformation

- unambiguous for relations representing 1:N
- 1:1 relationships: merge with one of the "entity-tables"
- M:N relationships: never merge
  Representation in RDM always by a separate table!

Always merge 1:N relationships?

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

```
Person(id, name, rNo, since, netSocket#...)
```


E-R to RDM mapping: Simplification

Merging 1:N relationships

- makes sense in most cases
- if relationship has many attributes do not merge if many NULL values expected
- if attributes of relationship are used infrequently by applications, do not merge
  This is an efficiency argument: avoid unnecessary data transfers

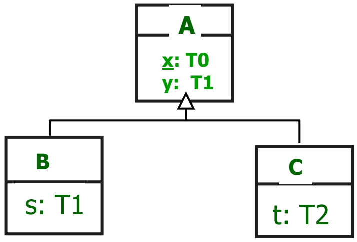
Never merge M:N relationships

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

~~⇨ Person(id, name,..., hobby, casualty...)~~

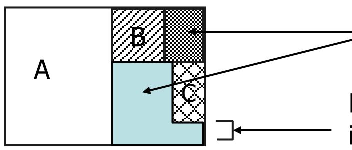
~~Key changed, redundancy~~

3.3.3 E-R to RDM mapping: Generalization



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

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



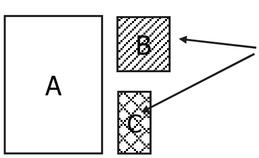
NULLs, if subtypes are disjoint

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

E-R to RDM mapping: Generalization

Second Alternative:

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



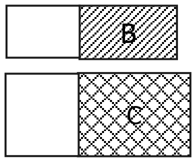
B and C are separate relations
In this example:
- disjoint specializations
- not exhaustive

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

Key of A as foreign key in B and C

E-R to RDM mapping: Generalization

Third alternative



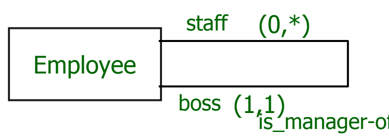
AB(x,y,s)
AC(x,y,t)

If subtypes are exhaustive, separate tables which include A's attributes, are a reasonable choice.

First solution: may be many Null values
Second solution: joining data from different tables (A and B or C) to get all about C or B may be time consuming
Third solution: only valid for complete specializations

E-R to RDM mapping

Recursive relationships



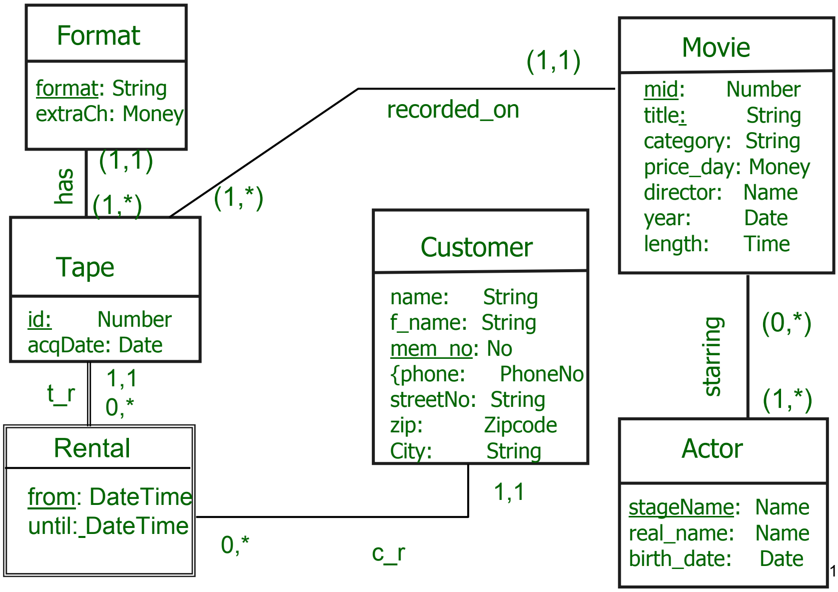
Employee (eid, …., managed_by_eid,….)

– Transformation step depending on cardinalities just like non-recursive relationships

N-ary relationships

– Like binary N:M relationships: Transform into one table with a composite key from the entities involved and attributes from the relationship, if any

Case study revised



Logical Design E-R to RDM mapping

```
Tape(id, m_id, format, acqDate )
Movie(id, title, category, price_day,
      director, year, length)
Customer(mem_no,name,f_name, zip,
         city, streetNo)
Actor(stage_name, real_name, birth_date)
Format(format, extraCh)
Rental (t_Id,from, until)
Phones (phoneNo, mem_no)
Starring (stageName, movieId)
```

Summary ER to RDM (system independent)

- Represent entities by relations (tables)
- Treat weak entities and generalization in a special way
- Represent relationships by relations, keys depend on cardinality of relationship
- Simplify the abstract schema by folding relations having the same key .

Note:

- Lost nearly all constraints in the abstract relational schema
  - Most cardinalities are lost
  - No existential dependencies (as for weak entities)
  - No value restrictions for attributes (not part of E-R)
  - Key constraints survived
- Concrete SQL / DDL allows to specify most of them

Summary

- Relational data model
  - Representation of data as relations(tables)
  - Very simple strucuture has pros and cons (which?)
  - the most important data model today
- Important terms & concepts
  - Relation: set of n-tuples
  - Relation schema defines structure
  - Attribute: property of relation, atomic values
  - Superkey, candidate key, primary key identify tuple
  - Transformation rules for entities, relationships
  - Simplification