DATABASTEKNIK - 1DL116

Fall 2003

An introductory course on database systems

http://user.it.uu.se/~udbl/dbt-ht2003/

Kjell Orsborn
Uppsala Database Laboratory
Department of Information Technology, Uppsala University,
Uppsala, Sweden

Introduction to the Relational Model

Elmasri/Navathe ch 7, 9.1

Kjell Orsborn

Department of Information Technology
Uppsala University, Uppsala, Sweden

The Relational Model

- The relational model was introduced by E. F. Codd 1970.
- Many DBMS's are based on this data model.
- It support simple declarative, but yet powerful, languages for describing operations on data.
- Operations in the relational model applies to relations (tables) and produce new relations.
  - This means that an operation can be applied to the result of another operation and that several different operations can be combined.
  - Operations are described in an algebraic notation that is based on relational algebra.

Relations as mathematical objects

- In set theory, a relation is defined as a subset of the product set (cartesian product) of a number of domains (value sets).
- The product set of the domains $D_1, D_2, \ldots, D_n$ is written as $D_1 \times D_2 \times \ldots \times D_n$.
- $\mathbf{D_1 \times D_2 \times \ldots \times D_n}$ constitute the set of all ordered sets $<v_1, v_2, \ldots, v_n>$ such that v_i belongs to D_i for all i.
    - If n=2, $D_1=\{T, F\}$ and $D_2=\{P, Q, R\}$ one gets the product sets:
      $D_1 \times D_2 = \{<T,P>,<T,Q>,<T,R>,<F,P>,<F,Q>,<F,R>\}$
      $D_2 \times D_1 = \{<P,T>,<P,F>,<Q,T>,<Q,F>,<R,T>,<R,F>\}$
    - For example, we have the relations:
      $R_1 \subseteq D_2 \times D_1 \qquad R_1= \{<P,T>,<Q,T>,<R,T>\}$
      $R_2 \subseteq D_2 \times D_1 \qquad R_2= \{<P,T>,<P,F>\}$
- Members of a relation is called **tuples**. If the relation is of **degree** n, the tuples are called *n-tuples*.

An example relation

- If
  customer-name = {Jones, Smith, Curry, Lindsay }
  customer-street = { Main, North, Park }
  customer-city = { Harrison, Rye, Pittsfield }
- Then
  r = {(Jones, Main, Harrison), (Smith, North, Rye), (Curry, North, Rye), (Lindsay, Park, Pittsfield)}
  is a relation over *customer-name* × *customer-street* × *customer-city*

Relation schema

- $A_1, A_2, \ldots, A_n$ are attributes
- $R = (A_1, A_2, \ldots, A_n)$ is a relation schema
  - *Customer-schema(customer-name, customer-street, customer-city)*
- r(R) is a relation on the relation schema R
  - *customer (Customer-schema)*

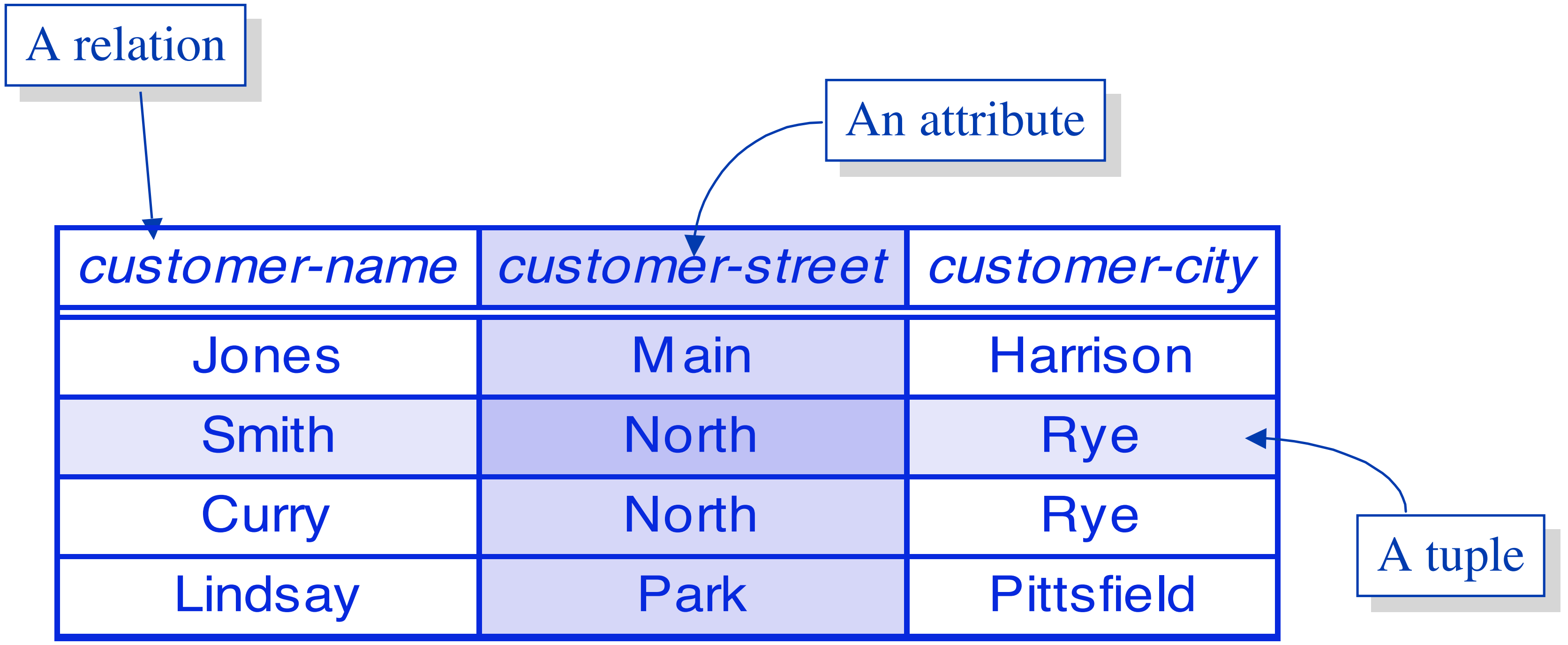
Relation instance

- The current values (*relation instance*) of a relation are specified by a table.
- An element *t* of *r* is a tuple - represented by a *row* in a table customer

customer

customer-name	*customer-street*	*customer-city*
Jones	Main	Harrison
Smith	North	Rye
Curry	North	Rye
Lindsay	Park	Pittsfield

Relations as tables



customer-name	*customer-street*	*customer-city*
Jones	Main	Harrison
Smith	North	Rye
Curry	North	Rye
Lindsay	Park	Pittsfield

First Normal Form

- Only simple or atomic values are allowed in the relational model.
- Attributes is not allowed to have composite or multiple values.
- The theory for the relational model is based on these assumptions which is called:

The first normal form assumption

Null values

- A special value, **null** or □, can sometimes be used as an attribute value.
- Every occurence of null is unique. Thus, two occurences of null is not considered to be equal even if they are represented by the same symbol.
- null is used:
  - when one does not know the actual value of an attribute.
  - when a certain attribute does not have a value.
  - when an attribute is not applicable.
- Examples of the use of null are showed later.

Keys

- Because relations are sets, all tuples in the relation are different.
- There is usually a subset k of the attributes in a relation schema R, i.e. k □ R, that has the characteristic that if the tuples
  t1, t2 □ r(R) and t1 ≠ t2, the following holds:
  t1[k] ≠ t2[k] (i.e. the value of k in t1 ≠ the value of k in t2)
- Every such subset k is called a **superkey** for R.

Keys - continued . . .

- A superkey k is *minimal* if there is no other superkey k' such that k' □ k.
- Every minimal superkey (NOTE! there can be more than one) is called a **candidate key** for R.
- The candidate key chosen by the database designer as the key for R is called R:s **primary key** or just **key**.
- In addition, term **foreign key** is used when a tuple is referenced, from another relation, with its key.

Key examples

- Example superkey:
  - {customer-name, customer- street} and {customer- name} are both superkeys of *Customer*, if no two customers can possibly have the same name.
- Example candidate key:
  - {customer- name} is a candidate key for *Customer* , since it is a superkey (assuming no two customers can possibly have the same name), and no subset of it is a superkey.

Determining keys from E-R types

- **Strong entity type**. The primary key of the entity type becomes the primary key of the relation.
- **Weak entity type**. The primary key of the relation consists of the union of the primary key of the strong entity type and the discriminator of the weak entity type.
- **Relationship type**. The union of the primary keys of the related entity types becomes a super key of the relation.
  - For binary many-to-many relationship types, above super key is also the primary key.
  - For binary many-to-one relationship types, the primary key of the "many" entity type becomes the relation's primary key.
  - For one-to-one relationship types, the relation's primary key can be that of either entity type.

Integrity constraints

for a relational database schema

- 1. Domain constraint
  - attribute values for attribute A shall be atomic values from dom(A)
- 2. Key constraint
  - candidate keys for a relation must be unique
- 3. Entity integrity constraint
  - no primary key is allowed to have a null value
- 4. Referential integrity constraint
  - a tuple that refers to another tuple in another relation must refer to an existing tuple
- 5. Semantic integrity constraint
  - e.g. “an employee’s total work time per week can not exceed 40 hours for all projects taken all together”

From E-R to relational model

- The basic procedure defines a set of relational schemas that represent entity and relationship types in the E-R model. This model should further with integrity constraints.
  - Primary keys allow entity types and relationship types to be expressed uniformly as *tables* which represent the contents of the database.
  - A database which conforms to an E-R diagram can be represented by a collection of tables.
  - For each entity type and relationship type there is a unique table which is assigned the name of the corresponding entity type or relationship type.
  - Each table has a number of columns (generally corresponding to attributes), which have unique names.
  - Converting an E-R diagram to a table format is the basis for deriving a relational database design from an E-R diagram.

Steps in translation from E-R model to relational model

- Translation of entity types and their attributes
  - Step 1) Entity types
  - Step 2) Weak entity types
- Translation of relationships
  - Step 3) 1-1 Relationship
  - Step 4) 1-N Relationship
  - Step 5) M-N Relationship
- Translation of multivalued attributes and relationships
  - Step 6) Multivalued attributes
  - Step 7) Multivalued relationships

Translating entity types and their attributes

- Step 1: Entity types - a strong entity type reduces to a table with the same attributes.
  - Key attributes (primary key - pk) is made the primary key column(s) for the table. Each attribute gets their own column.
  - Composite attributes are normally represented by their simple components.
  - Example customer schema and table:

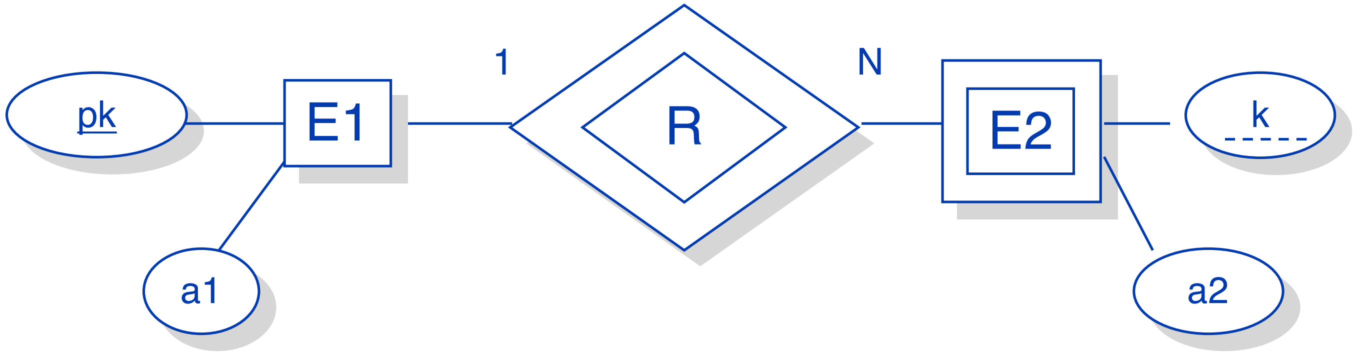
pk

Customer(<u>social-security</u>, customer-name, c-street, c-city)

<u>social-security</u>	*customer-name*	*c-street*	*c-city*
321-12-3123	Jones	Main	Harrison
019-28-3746	Smith	North	Rye
677-89-9011	Hayes	Main	Harrison

Translating entity types cont. . .

- Step 2: **Weak entity types** - a weak entity type becomes a table that includes a column for the primary key of the identifying strong entity type .



pk	a1

pk	k	a2

UPPSALA UNIVERSITET

Translating entity types cont. . .

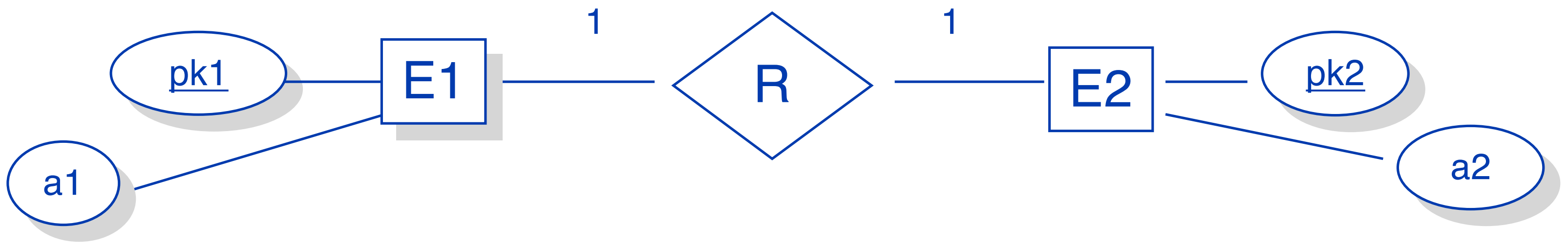
- The table corresponding to a relationship type linking a weak entity type to its identifying strong entity type is redundant.
- Example of the payment schema and table:
  - The payment table already contains the information that would appear in the loan-payment table (i.e., the columns loan-number and payment-no).

Payment(loan-number, payment-no, pay-date, amount)

loan-number	*payment-no*	*pay-date*	*amount*
L-17	5	10 May 1996	50
L-23	11	17 May 1996	75
L-15	22	23 May 1996	300

Translating relationship types

- Step 3: 1-1 Relationship types
  - The foreign key column (fk) is a copy of the other entity’s primary key column (pk). The values in a fk-column point to unique row in the other table, and thus implement the relationship.



Alt 1:

pk1	a1

pk2	a2	f k1

Alt 2:

pk1	a1	f k2

pk2	a2

Translating 1-1 relationship types cont. . .

Alt 3:

E1

pk1	a1

R

f k1	f k2

E2

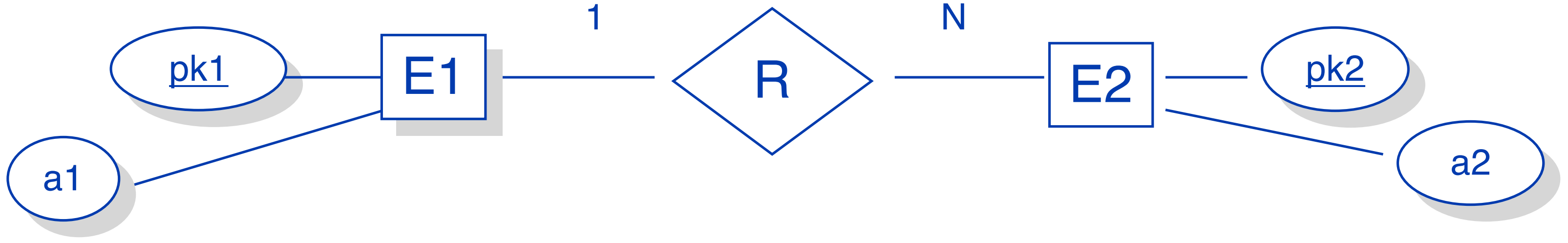
pk2	a2

Alt 4:

E1		E2	
pk1	a1	pk2	a2

Translating relationship . . . cont. . .

- Step 4: 1-N Relationship types
  - Include the primary key of the "1-side" as a foreign key on the "N-side", (i.e. the foreign key column is placed on the entity on the N-side).
  - Alternatively, an extra table (R) is created whose primary key is a foreign key composed by the primary key from the N-side.



Alt 1:

pk1	a1

pk2	a2	f k1

Alt 2:

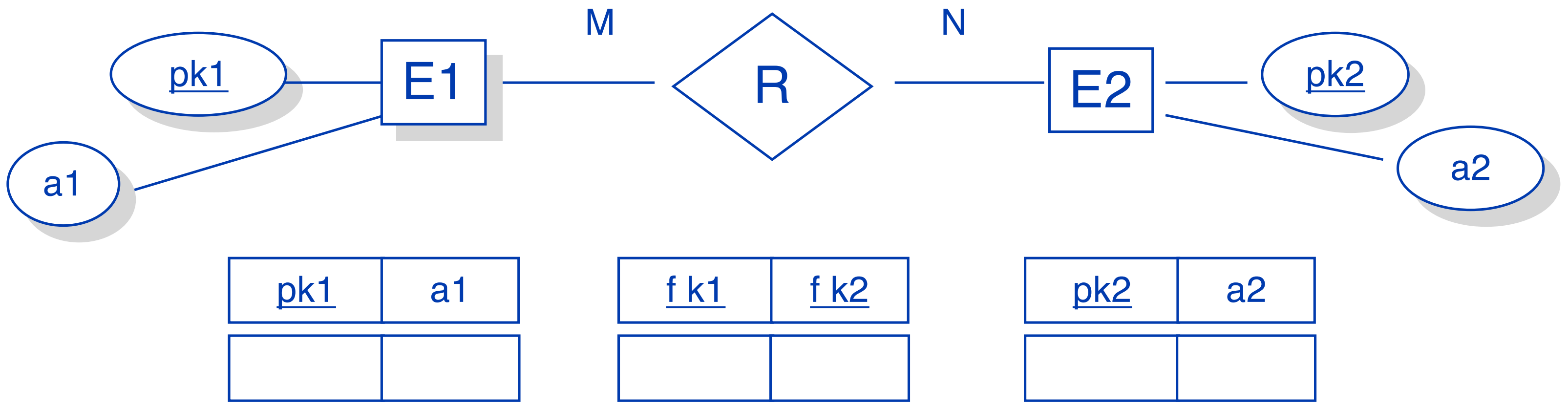
pk1	a1

f k1	f k2

pk2	a2

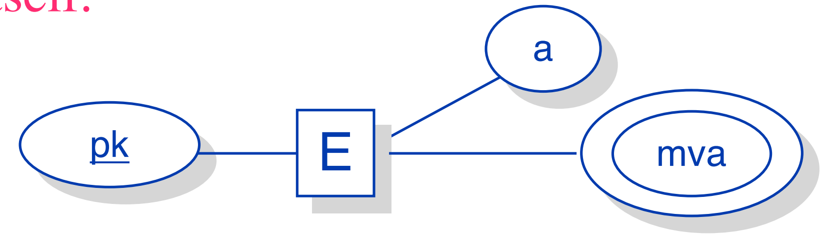
Translating relationship . . . cont. . .

- Step 5: M-N Relationship types
  - Always a separate table with columns for the primary keys of the two participating entity types, and any descriptive attributes of the relationship type.



Translating relationship . . . cont. . .

- Step 6: Multivalued attributes
  - A separate table is created for the multivalued attribute. Its primary key is composed of the owning entity's primary key, and the attribute value itself.



E

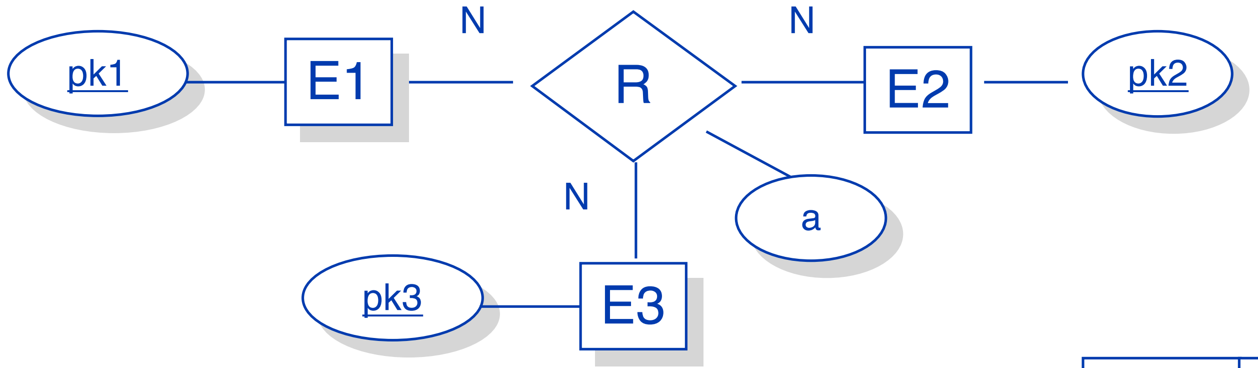
pk	a

E-MVA

pk	mva

Translating relationship . . . cont. . .

- Step 7: Multivalued relationship types
  - First try to remove multivalued relationships on the E-R model level by model transformation.
  - A separate table is created, with foreign keys to all tables that are included in the relationship. Its primary key is composed of all foreign keys.



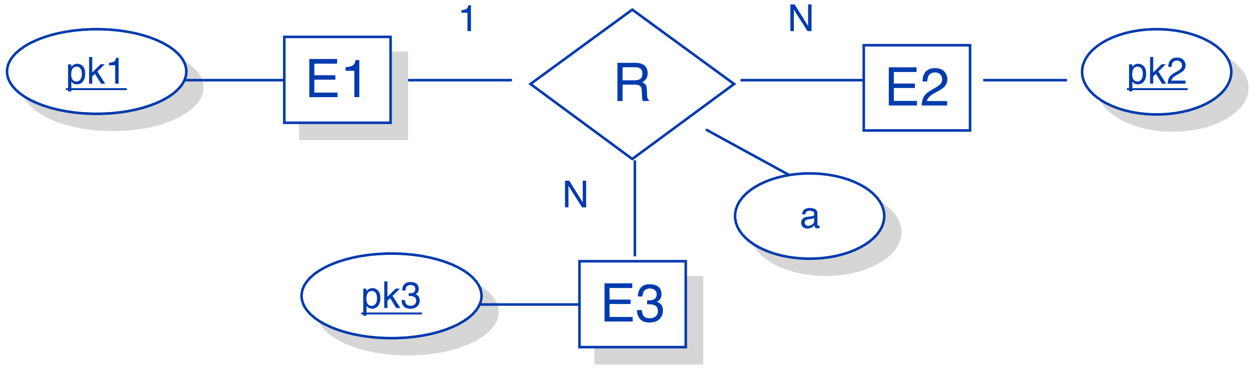
R

f k1	f k2	f k3	a

UPPSALA UNIVERSITET

Translating relationship . . . cont. . .

- Step 7: Multivalued relationship types continued
  - In the case where R is 1-N-N, the primary key on R shall not include the fk for the table with cardinality 1.



R

f k1	<u>f k2</u>	<u>f k3</u>	a

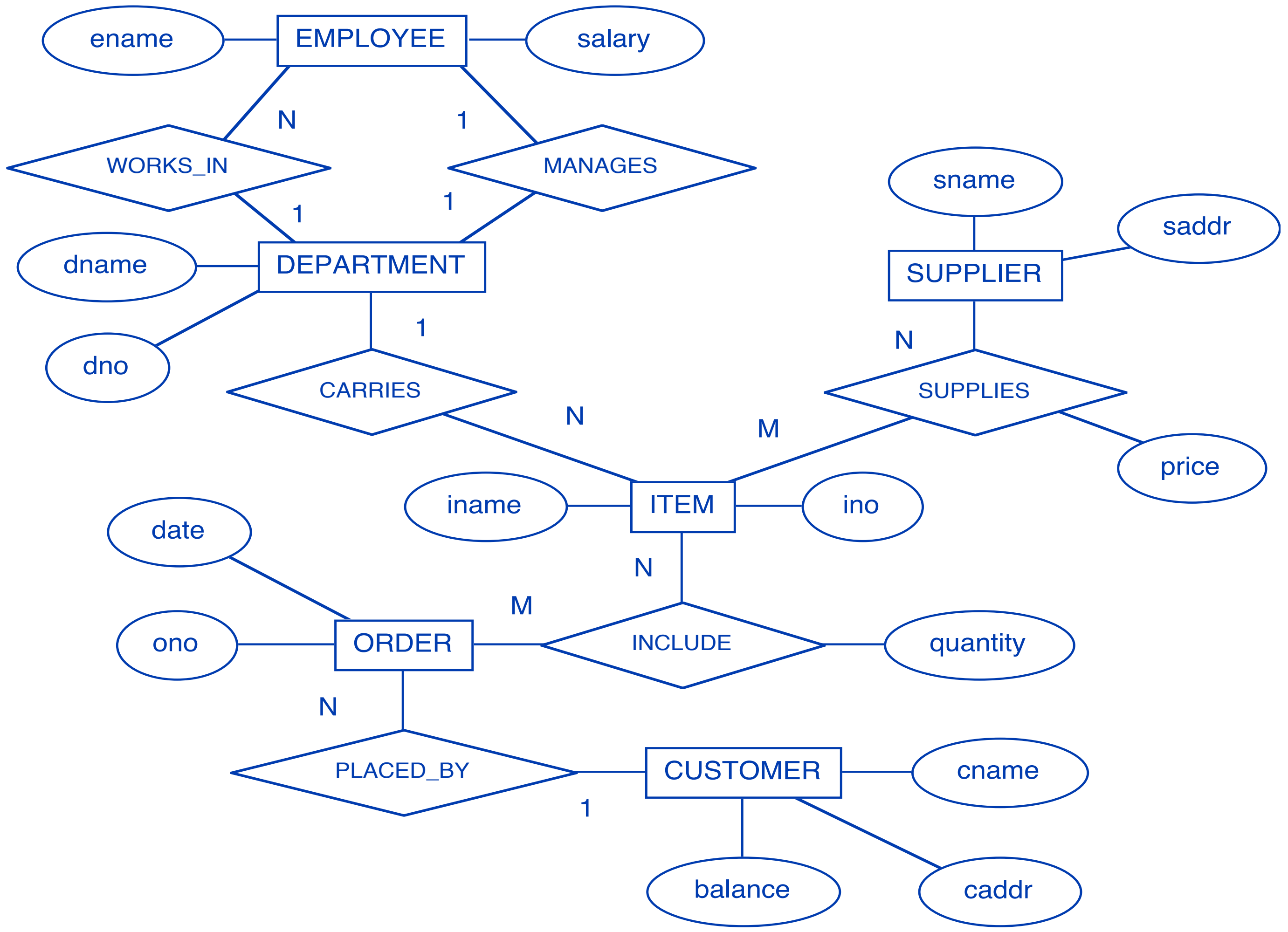
Summary

- Entity types and their attributes
  - Step 1) Entity types
    - Each entity gets a corresponding table, with the primary key column set to its key attribute.
  - Step 2) Weak entity types
    - The primary key of a weak entity type table has the primary key of the owner table as a component.
- Relationships
  - Step 3) 1-1 Relationship
    - 4 alternatives: fk in E1 or E2, separate R table, common table for E1 & E2
  - Step 4) 1-N Relationship
    - fk i entity on the N-side, separate R table
  - Step 5) M-N Relationship
    - separate R table

Summary cont. . .

- Multivalued attributes and relationships
  - Step 6) Multivalued attributes
    - Separate table for the attribute with its pk composed of the owner pk and the value column.
  - Step 7) Multivalued relationships
    - Separate R table. N-N-N: pk composed of all fk’s. 1-N-N: pk is fk to the E1-table.

Example E-R to relational model translation



Relational schemas for the example

- Schemas for the entity types in the example above

    EMP(<u>ENAME</u>, SALARY, DEPT)

    DEPTS(<u>DNAME</u>, DEPT#, MGR)

    SUPPLIERS(<u>SNAME</u>, SADDR)

    ITEMS(<u>INAME</u>, ITEM, DNAME)

    ORDERS(<u>O#</u>, DATE, CUST)

    CUSTOMERS(<u>CNAME</u>, CADDR, BALANCE)

- Schemas for relationship types (M:N)

    SUPPLIES(<u>SNAME</u>, <u>INAME</u>, PRICE)

    INCLUDES(<u>O#</u>, <u>INAME</u>, QUANTITY)

Short summary E-R -> R

E-R concept	Relational concept
entity type	relation
1:1 relationship type	include one of the primary keys as a foreign key of the other ”entity relation”
1:N relationship type	include the ”1-side” primary key as a foreign key at the ”n-side”
M:N relationship type	relation with two foreign keys
n-ary relationship type (degree > 2)	relation with n foreign keys
simple attribute	attribute
composite attribute	simple attribute components
multivalued attribute	relation anf foreign key
value set	domain
key attribute	primary (or secondary key)

Introduction to Relational Algebra

Elmasri/Navathe ch 7

Kjell Orsborn

Department of Information Technology
Uppsala University, Uppsala, Sweden

Query languages

- Languages where users can express what information to retrieve from the database.
- Categories of query languages:
  - Procedural
  - Non-procedural (declarative)
- Formal ("pure") languages:
  - Relational algebra
  - Relational calculus
    - Tuple-relational calculus
    - Domain-relational calculus
  - Formal languages form underlying basis of query languages that people use.

Relational algebra

- **Relational algebra** is a procedural langaue
- Operations in relational algebra takes two or more relations as arguments and return a new relation.
- Relational algebraic operations:
  - Operations from set theory:
    - Union, Intersection, Difference, Cartesian product
  - Operations specifically introduced for the relational data model:
    - Select, Project, Join
- It have been shown that the *select, project, union, difference,* and *cartesian product* operations form a complete set. That is any other relational algebra operation can be expressed in these.

Operations from set theory

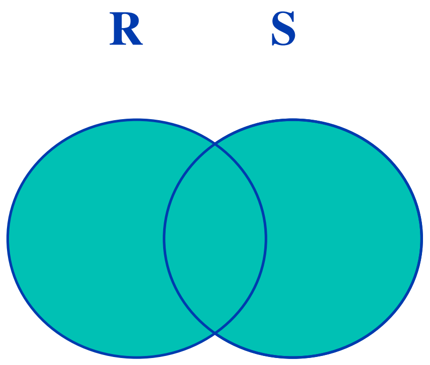
- Relations are required to be **union compatible** to be able to take part in the union, intersection and difference operations.
- Two relations R_1 and R_2 is said to be union-compatible if:

$R_1 \subseteq D_1xD_2x...xD_n$ and
$R_2 \subseteq D_1xD_2x...xD_n$

i.e. if they have the same degree and the same domains.

Union operation

- The **union** of two union-compatible relations *R* and *S* is the set of all tuples that either occur in *R*, *S*, or in both.
- Notation: $R \cup S$
- Defined as: $R \cup S = \{t \mid t \in R \text{ or } t \in S\}$
- For example:



A	*B*
a	1
a	2
b	1

$\cup$

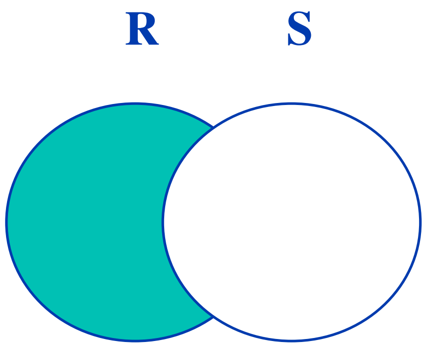
A	*B*
a	2
b	3

$=$

A	*B*
a	1
a	2
b	1
b	3

Difference operation

- The **difference** between two union-compatible sets *R* and *S* is the set of all tuples that occur in *R* but not in *S*.
- Notation: R □ S
- Defined as: R □ S = {t | t □ R and t □ S}
- For example:



A	B
a	1
a	2
b	1

□

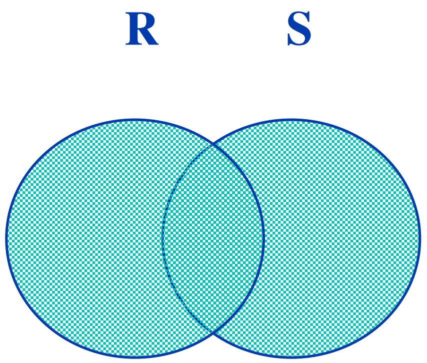
A	B
a	2
b	3

=

A	B
a	1
b	1

Intersection

- The **intersection** of two union-compatible sets *R* and *S*, is the set of all tuples that occur in both *R* and *S*.
- Notation: R □ S
- Defined as: R □ S = {t | t □ R and t □ S}
- For example:



A	B
a	1
a	2
b	1

□

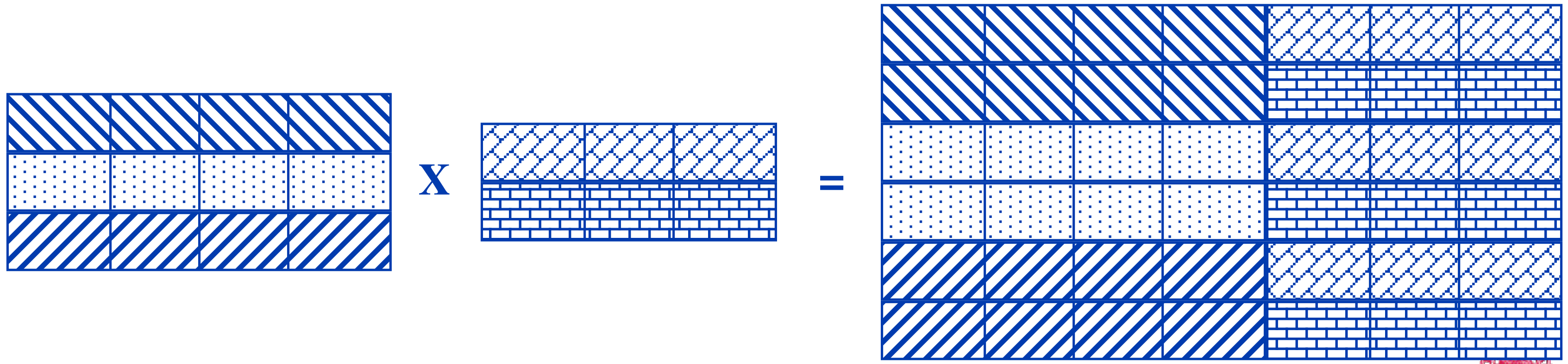
A	B
a	2
b	3

=

A	B
a	2

Cartesian product

- Let R and S be relations with k1 and k2 arities resp. The **cartesian product** of *R* and *S* is the set of all possible k_1+k_2 tuples where the first k_1 components constitute a tuple in *R* and the last k_2 components a tuple in *S*.
- Notation: R □ S
- Defined as: R □ S = {t q | t □ R and q □ S}
- Assume that attributes of r(R) and s(S) are disjoint. (i.e. R □ S = ∅). If attributes of r(R) and s(S) are not disjoint, then renaming must be used.



Cartesian product example

A	*B*
a	1
b	2

□

C	*D*
a	5
b	5
b	6
c	5

=

A	*B*	*C*	*D*
a	1	a	5
a	1	b	5
a	1	b	6
a	1	c	5
b	2	a	5
b	2	b	5
b	2	b	6
b	2	c	5

Selection operation

- The selection operator, □, selects a specific set of tuples from a relation according to a selection condition (or selection predicate) *P*.
- Notation: $□_{p}$(R)
- Defined as: $□_{p}$(R) = {t | t □ R and *P*(t) } (i.e. the set of tuples t in *R* that fulfills the condition *P*)
- Where *P* is a logical expression$^{(*)}$ consisting of terms connected by:
  □ (**and**), (**or**), □ (**not**)
  and each term is one of:
  <attribute> *op* <attribute> or <constant>
  where *op* is one of: =, ≠, >, ≥. <. □

  Example: $□_{\text{SALARY>30000}}$(EMPLOYEE)

  (*) a formula in propositional calculus

Selection example

R =

A	B	C	D
a	a	1	7
a	b	5	7
b	b	2	3
b	b	4	9

$\sigma_{A=B \wedge D>5}(R)$ =

A	B	C	D
a	a	1	7
b	b	4	9

Projection operation

- The **projection** operator, □, picks out (or projects) listed columns from a relation and creates a new relation consisting of these columns.
- Notation: □ $_{A_1,A_2,\ldots,A_k}$ (R)
  where A_1, A_2 are attribute names and R is a relation name.
- The result is a new relation of k columns.
- Duplicate rows removed from result, since relations are sets.

Example: □ $_{\text{LNAME,FNAME,SALARY}}$(EMPLOYEE)

Projection example

R =

A	*B*	*C*
a	1	1
a	2	1
b	3	1
b	4	2

$\square_{A,C}(R)$ =

A	*C*
a	1
~~a~~	~~1~~
b	1
b	2

=

A	*C*
a	1
b	1
b	2

Join operator

- The **join** operator, (almost), creates a new relation by joining related tuples from two relations.
- Notation: R $_C$ S
  C is the join condition which has the form A_r □ A_s , where □ is one of {=, <, >, ≤, ≥, ≠}. Several terms can be connected as C_1 □C_2 □...C_k.
- A join operation with this kind of general join condition is called “Theta join”.

Example Theta join

R

A	*B*	*C*
1	2	3
6	7	8
9	7	8

A≤D

S

B	*C*	*D*
2	3	4
7	3	5
7	8	9

=

R *A≤D* S

A	*B*	*C*	*B*	*C*	*D*
1	2	3	2	3	4
1	2	3	7	3	5
1	2	3	7	8	9
6	7	8	7	8	9
9	7	8	7	8	9

Equijoin

- The same as join but it is required that attribute A_r and attribute A_s should have the same value.
- Notation: R $_{C}$ S
  C is the join condition which has the form $A_r = A_s$. Several terms can be connected as C_1 □C_2 □...C_k.

Example Equijoin

R

A	*B*
a	2
a	4

B=C

S

C	*D*	*E*
2	d	e
4	d	e
9	d	e

=

R $_{B=C}$ S

A	*B*	*C*	*D*	*E*
a	2	2	d	e
a	4	4	d	e

Natural join

- **Natural join** is equivalent with the application of join to R and S with the equality condition $A_r = A_s$ (i.e. an equijoin) and then removing the redundant column A_s in the result.
- Notation: R $*_{Ar,As}$ S
  A_r,A_s are attribute pairs that should fulfil the join condition which has the form $A_r = A_s$. Several terms can be connected as $C_1 \square C_2 \square \ldots C_k$.

Example Natural join

R

A	B
a	2
a	4

$B{=}C$

S

C	D	E
2	d	e
4	d	e
9	d	e

=

$R \bowtie_{B=C} S$

A	B	D	E
a	2	d	e
a	4	d	e

Composition of operations

- Expressions can be built by composing multiple operations
- Example: $\sigma_{A=C}(R \times S)$

$R \times S$ =

A	B
a	1
b	2

$\times$

C	D
a	5
b	5
b	6
c	5

=

A	B	C	D
a	1	a	5
a	1	b	5
a	1	b	6
a	1	c	5
b	2	a	5
b	2	b	5
b	2	b	6
b	2	c	5

$\sigma_{A=C}(R \times S)$ =

A	B	C	D
a	1	a	5
b	2	b	5
b	2	b	6

Additional relational operations

- Assignment and Rename
- Division
- Outer join and outer union
- Aggregate functions (presented together with SQL)
- Update operations (presented together with SQL)
  - (not part of pure query language)

Assignment operation

- The assignment operation ($\leftarrow$) makes it possible to assign the result of an expression to a temporary relation variable.
- Example:
- $temp \leftarrow \sigma_{dno\ =\ 5}(EMPLOYEE)$

  $result \leftarrow \Pi_{fname,lname,salary}\ (temp)$
- The result to the right of the $\leftarrow$ is assigned to the relation variable on the left of the $\leftarrow$.
- The variable may use variable in subsequent expressions.

Renaming relations and attribute

- The assignment operation can also be used to rename relations and attributes.
- Example:
  NEWEMP □ $□_{\text{dno = 5}}$(EMPLOYEE)
  R(FIRSTNAME,LASTNAME,SALARY) □
  $\Pi_{\text{fname,lname,salary}}$ (NEWEMP)

Division operation

- Suited to queries that include the phrase “for all”.
- Let R and S be relations on schemas R and S respectively, where $R = (A_1,\ldots,A_m, B_1,\ldots,B_n)$
  $S = (B_1,\ldots,B_n)$
- The result of R ÷ S is a relation on schema
  $R - S = (A_1,\ldots,A_m)$
  $R \div S = \{t \mid t \in \Pi_{R-S}(R) \land \forall u \in S\,(tu \in R)\}$

Example Division operation

R

A	*B*
a	1
a	2
a	3
b	1
c	1
d	1
d	3
d	4
d	6
e	1
e	2

÷

S

B
1
2

=

R ÷ S

A
a
e

Outer join/union operation

- Extensions of the join/union operations that avoid loss of information.
- Computes the join/union and then adds tuples from one relation that do not match tuples in the other relation to the result of the join.
- Fills out with *null* values:
  - *null* signifies that the value is unknown or does not exist.
  - All comparisons involving null are **false** by definition.

Example Outer join

- Relation *loan*

branch-name	*loan-number*	*amount*
Downtown	1-170	3000
Redwood	L-230	4000
Perryridge	L-260	1700

- Relation *borrower*

customer-name	*loan-number*
Jones	1-170
Smith	L-230
Hayes	L-155

Example Outer join cont...

- *loan * borrower* (natural join)

branch-name	*loan-number*	*amount*	*customer-name*
Downtown	1-170	3000	Jones
Redwood	L-230	4000	Smith

- *loan* left *borrower* (left outer join)

branch-name	*loan-number*	*amount*	*customer-name*	*loan-number*
Downtown	1-170	3000	Jones	1-170
Redwood	L-230	4000	Smith	L-230
Perryridge	L-260	1700	*null*	*null*

Example Outer join cont...

- *loan* $_{\text{right}}$ *borrower* (natural right outer join)

branch-name	*loan-number*	*amount*	*customer-name*
Downtown	L-170	3000	Jones
Redwood	L-230	4000	Smith
null	L-155	*null*	Hayes

- *loan* $_{\text{full}}$ *borrower* (natural full outer join)

branch-name	*loan-number*	*amount*	*customer-name*
Downtown	L-170	3000	Jones
Redwood	L-230	4000	Smith
Perryridge	L-260	1700	*null*
null	L-155	*null*	Hayes

Aggregation operations

- Presented together with SQL later
- Examples of aggregation operations
  - avg
  - min
  - max
  - sum
  - count

Update operations

- Presented together with SQL later
- Operations for database updates are normally part of the DML
  - **insert** (of new tuples)
  - **update** (of attribute values)
  - **delete** (of tuples)
- Can be expressed by means of the assignment operator

Example DB schema

- In the following example we will use a database with the following relation schemas:
  - emps(ename, salary, dept)
  - depts(dname, dept#, mgr)
  - suppliers(sname, addr)
  - items(iname, item#, dept)
  - orders(o#, date, cust)
  - customers(cname, addr, balance)

  - supplies(sname, iname, price)
  - includes(o#, item, quantity)

Relation algebra as a query language

- Relational schema: supplies($\underline{\text{sname}}$, $\underline{\text{iname}}$, price)
- “What is the names of the suppliers that supply cheese?”

  $\pi_{\text{sname}}(\sigma_{\text{iname='CHEESE'}}(\text{SUPPLIES}))$
- “What is the name and price of the items that cost less than 5 \$ and that are supplied by WALMART”

  $\pi_{\text{iname,price}}(\sigma_{\text{sname='WALMART'} \wedge \text{price} < 5}(\text{SUPPLIES}))$