Combining Relations

- **Example:** Let D and S be relations on A = {1, 2, 3, 4}.
- D = {(a, b) | b = 5 - a}   "b equals (5 – a)"
- S = {(a, b) | a < b}   "a is smaller than b"

- D = {(1, 4), (2, 3), (3, 2), (4, 1)}
- S = {(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)}
- S∘D = { **(2, 4), (3, 3), (3, 4), (4, 2), (4, 3), (4, 4)}**

D maps an element a to the element (5 – a), and afterwards S maps (5 – a) to all elements larger than (5 – a), resulting in S∘D = {(a,b) | b > 5 – a} or S∘D = {(a,b) | a + b > 5}.

Combining Relations

- We already know that **functions** are just **special cases** of **relations** (namely those that map each element in the domain onto exactly one element in the codomain).

- If we formally convert two functions into relations, that is, write them down as sets of ordered pairs, the composite of these relations will be exactly the same as the composite of the functions (as defined earlier).

Combining Relations

- **Definition:** Let R be a relation on the set A. The powers R^n, $n = 1, 2, 3, \ldots$, are defined inductively by
- $R^1 = R$
- $R^{n+1} = R^n \circ R$

- In other words:
- $R^n = R \circ R \circ \ldots \circ R$ (n times the letter R)

Combining Relations

- **Theorem:** The relation R on a set A is transitive if and only if $R^n \subseteq R$ for all positive integers n.
- Remember the definition of transitivity:
- **Definition:** A relation R on a set A is called transitive if whenever $(a, b) \in R$ and $(b, c) \in R$, then $(a, c) \in R$ for $a, b, c \in A$.
- The composite of R with itself contains exactly these pairs (a, c).
- Therefore, for a transitive relation R, $R \circ R$ does not contain any pairs that are not in R, so $R \circ R \subseteq R$.
- Since $R \circ R$ does not introduce any pairs that are not already in R, it must also be true that $(R \circ R) \circ R \subseteq R$, and so on, so that $R^n \subseteq R$.

n-ary Relations

- In order to study an interesting application of relations, namely **databases**, we first need to generalize the concept of binary relations to **n-ary relations**.

- **Definition:** Let $A_1, A_2, \ldots, A_n$ be sets. An **n-ary relation** on these sets is a subset of $A_1 \times A_2 \times \ldots \times A_n$.
- The sets $A_1, A_2, \ldots, A_n$ are called the **domains** of the relation, and n is called its **degree**.

n-ary Relations

- **Example:**
- Let $R = \{(a, b, c) \mid a = 2b \wedge b = 2c \text{ with } a, b, c \in \mathbf{N}\}$
- What is the degree of R?
- The degree of R is 3, so its elements are triples.
- What are its domains?
- Its domains are all equal to the set of integers.
- Is (2, 4, 8) in R?
- No.
- Is (4, 2, 1) in R?
- Yes.

Databases and Relations

- Let us take a look at a type of database representation that is based on relations, namely the **relational data model.**

- A database consists of n-tuples called **records**, which are made up of **fields**.
- These fields are the **entries** of the n-tuples.

- The relational data model represents a database as an n-ary relation, that is, a set of records.

Databases and Relations

- **Example:** Consider a database of students, whose records are represented as 4-tuples with the fields **Student Name**, **ID Number**, **Major**, and **GPA**:

- R = {(Ackermann, 231455, CS, 3.88),
  (Adams, 888323, Physics, 3.45),
  (Chou, 102147, CS, 3.79),
  (Goodfriend, 453876, Math, 3.45),
  (Rao, 678543, Math, 3.90),
  (Stevens, 786576, Psych, 2.99)}

- Relations that represent databases are also called **tables**, since they are often displayed as tables.

Databases and Relations

- A domain of an n-ary relation is called a **primary key** if the n-tuples are uniquely determined by their values from this domain.
- This means that no two records have the same value from the same primary key.
- In our example, which of the fields **Student Name**, **ID Number**, **Major**, and **GPA** are primary keys?
- **Student Name** and **ID Number** are primary keys, because no two students have identical values in these fields.
- In a real student database, only **ID Number** would be a primary key.

Databases and Relations

- In a database, a primary key should remain one even if new records are added.
- Therefore, we should use a primary key of the **intension** of the database, containing all the n-tuples that can ever be included in our database.

- **Combinations of domains** can also uniquely identify n-tuples in an n-ary relation.
- When the values of a **set of domains** determine an n-tuple in a relation, the **Cartesian product** of these domains is called a **composite key**.

Databases and Relations

- We can apply a variety of **operations** on n-ary relations to form new relations.

- **Definition:** The **projection** $P_{i_1, i_2, \ldots, i_m}$ maps the n-tuple $(a_1, a_2, \ldots, a_n)$ to the m-tuple $(a_{i_1}, a_{i_2}, \ldots, a_{i_m})$, where $m \leq n$.

- In other words, a projection $P_{i_1, i_2, \ldots, i_m}$ keeps the m components $a_{i_1}, a_{i_2}, \ldots, a_{i_m}$ of an n-tuple and deletes its (n – m) other components.

- **Example:** What is the result when we apply the projection $P_{2,4}$ to the student record (Stevens, 786576, Psych, 2.99) ?
- **Solution:** It is the pair (786576, 2.99).

Databases and Relations

- In some cases, applying a projection to an entire table may not only result in fewer columns, but also in **fewer rows**.

- Why is that?

- Some records may only have differed in those fields that were deleted, so they become **identical**, and there is no need to list identical records more than once.

Databases and Relations

- We can use the **join** operation to combine two tables into one if they share some identical fields.

- **Definition:** Let R be a relation of degree m and S a relation of degree n. The **join** $J_p(R, S)$, where $p \leq m$ and $p \leq n$, is a relation of degree $m + n - p$ that consists of all $(m + n - p)$-tuples
$(a_1, a_2, \ldots, a_{m-p}, c_1, c_2, \ldots, c_p, b_1, b_2, \ldots, b_{n-p})$,
where the m-tuple $(a_1, a_2, \ldots, a_{m-p}, c_1, c_2, \ldots, c_p)$ belongs to R and the n-tuple $(c_1, c_2, \ldots, c_p, b_1, b_2, \ldots, b_{n-p})$ belongs to S.

Databases and Relations

- In other words, to generate $J_p(R, S)$, we have to find all the elements in R whose p last components match the p first components of an element in S.

- The new relation contains exactly these matches, which are combined to tuples that contain each matching field only once.

Databases and Relations

- **Example:** What is $J_1(Y, R)$, where Y contains the fields **Student Name** and **Year of Birth**,
- Y = {(1978, Ackermann),
  (1972, Adams),
  (1917, Chou),
  (1984, Goodfriend),
  (1982, Rao),
  (1970, Stevens)},
- and R contains the student records as defined before ?