Relational Databases - Comp2400 / Comp6240

Lecture 5: Sets, Products, Relations, Functions

Maths is more than numbers

The mathematical ideas that relational databases are built on.

- Sets, indexed sets
- Products
  (a more general definition than the usual one)
- Relations
- Functions

This will prepare us for some relational algebra, and normalisation, which is largely about "functional dependencies".

Sets

- a *set* is several things considered together as one thing
- there are two ways of specifying a set:
  1. list the items in it
     - $\{Sydney, Toulouse, London\}$
     - $\{2, 3, 15, 328\}$
     - $\{\}$
     - $\{\{1\}, \{1, 2\}, \{1, 2, 3\}, ...\}$
  2. describe its members using a property (or "constraint")
     - { the wheels on Greg's car }
     - { students currently enrolled in Comp6240 }
     - { goldfish currently enrolled in Comp6240 }
     - { the finite sets of integers }
- a set's members have no order: $\{1, 2\} = \{2, 1\}$
- things can not be in the set more than once: $\{1, 1\} = \{1\}$

Set operations and relations

membership	we write $x \in \{x, y, z\}$ to say that x is in the set
subset	if every member of A is also in B we write $A \subseteq B$
equality	two sets are equal if they have the same members
union	we write $A \cup B$ for the set containing everything in A and everything in B
intersection	we write $A \cap B$ for the set of things that are in both A and B
subtraction	$A - B$ is the elements from A that are *not* in B

Exercise: Set operations

Let $A = \{1, 2, 3\}$ and $B = \{true, false\}$.
Which of the following are correct?

1. $\{2\} \subseteq A$
2. $2 \in B$
3. $\{2, 3\} \in (A \cup B)$
4. $2 \in (A \cap B)$
5. $2 \in (A - \{1, 3\})$

Ordered Pairs and *n*-tuples

- an ordered pair is two things in order:
  a first thing and a second thing
  - (15, "turnip")
  - (*Melbourne*, *true*)
- the order matters: $(1, 2) \neq (2, 1)$
- the same thing can be in a pair twice:
  $(1, 1)$ is an ordered pair
- pairs are 2-tuples, we can have n-tuples for any positive integer n

No More Equivocation!

Database theory is schizophrenic about what a relation actually is.

Are attributes addressed by name or by position? Are the names just synonyms for positions?

The following slightly more sophisticated definition of relation accounts for both.

It also yields neat and uniform definitions of the main database ideas.

Indexed Sets

Sometimes its useful to give names to the elements of a set.

Here's an example where the names are numbers

$$A = \{\{1\}, \{1, 2\}, \{1, 2, 3\}, ...\}$$

$$A_n = \{1, ..., n\}$$
$$A = \{A_1, A_2, A_3, ...\}$$

We say that A is *indexed by* the positive integers $\mathbb{Z}^+$.
(do not confuse this with indexed files in computer systems)

Indexed Sets

The set of names is called the *index set*.

An indexed set is really a *function* from the index set into the set it indexes. In the previous example, A is a function that takes a positive integer n to the set $\{1, \ldots, n\}$.

Sometimes we will write eg. $\{fred = x, joe = y\}$
for $A = \{x, y\}$ where $A_{fred} = x$ and $A_{joe} = y$.

What is the index set in this example?

Tuples are Indexed Sets

We can think of ordered n-tuples as sets indexed by $\{1, ..., n\}$.

For example
$A = (x, y)$ can be seen as
$A = \{x, y\}$ with $A_1 = x$ and $A_2 = y$.

Problem? What about $(5, 5, 5)$?

No problem: $A = \{5\}$, $A_1 = A_2 = A_3 = 5$.
The same element can have several names.

Products - Simple Version

The product operation takes an ordered pair of sets,
and gives a set of ordered pairs.

It is all the pairs with the first thing from the first set, and the second thing from the second set.

$$A \times B = \{\text{ordered pairs } (a, b) \text{ where } a \in A \text{ and } b \in B\}$$

eg.

$$\{Sydney, Paris\} \times \{2, 4, 11\} =$$
$$\{(Sydney, 2), (Sydney, 4), (Sydney, 11),$$
$$(Paris, 2), (Paris, 4), (Paris, 11)\}$$

Products in queries: `select * from A, B;`

Products - General Version

The product operation takes an indexed set of sets,
and gives a set of indexed sets.

It is all the indexed sets with the i-thing from the i-set, for each i in the index set.

$$\prod_{i \in I} A_i = \{\text{sets } \alpha \text{ indexed by I where each } \alpha_i \in A_i\}$$

eg. Let $A_{city} = \{Sydney, Paris\}$ and $A_{custID} = \{2, 4, 11\}$, then

$$\prod_{(i \in \{city, custID\})} A_i =$$

$$\{\{city = Sydney, custID = 2\}, \{city = Sydney, custID = 4\},$$
$$\{city = Sydney, custID = 11\}, \{city = Paris, custID = 2\},$$
$$\{city = Paris, custID = 4\}, \{city = Paris, custID = 11\}\}$$

Products - General Version Made Specific

Now we show that the general version really generalises the simple version

$$\begin{aligned} A_1 \times A_2 &= \prod_{i \in \{1,2\}} A_i \\ &= \{\text{sets } \alpha \text{ indexed by I where each } \alpha_i \in A_i\} \\ &= \{\text{sets } \{a_1, a_2\} \text{ where } a_1 \in A_1 \text{ and } a_2 \in A_2\} \\ &= \{\text{tuples } (a_1, a_2) \text{ where } a_1 \in A_1 \text{ and } a_2 \in A_2\} \end{aligned}$$

Relations

- a *relation schema* **R** is an indexed set of sets
  the index set is the attribute names N,
  it indexes the datatypes D
- a *relation* is a subset of a product
- the states of a relation schema are the subsets of the schema's product, that is R is an instance or state of **R** iff

$$R \subseteq \prod_{n \in N} D_n$$

- a database schema is an indexed set of relation schemas with some constraints
  (the index set is the relation schema names)

Functions

We mentioned functions already, but more formally

- a *function* $A \longrightarrow B$ is a relation $\subseteq A \times B$ which has exactly one pair (a, b) for each $a \in A$
- if f is a function $A \longrightarrow B$ and $(a, b) \in f$ then we say $f(a) = b$

Exercise: functions

Let $A = \{1, 2, 3\}$ and $B = \{true, false\}$.
Which of the following are functions $A \longrightarrow B$

1. $\{(1, true), (1, false)\}$
2. $\{(1, 2), (2, 3), (3, 1)\}$
3. $\{(1, true), (2, false), (3, true), (1, false)\}$
4. $\{(1, true), (2, false), (3, true), (1, true)\}$
5. $\{(1, true), (2, false), (3, true)\}$
6. $\{(1, true), (3, true)\}$

Labs

- Make sure you enrol in a lab group before next week
- Make sure you attend your lab (2% per lab, and important learning experience)
- Make sure you *understand* what you are doing - if stuck ask your tutor for help
- Remember: *no lecture* next Wednesday 6th August