CS109A Notes for Lecture 3/13/95

Database Relations

The *relational model* upon which modern database systems are based uses a slight variant of k-ary relations from set theory.

- Database relations are "tables" with
  - ☐ Columns headed by *attributes* (column names).
  - ☐ Rows called *tuples* that are members of the relation.

Class	Weight	Guns	Caliber	Type	Country
Alaska	28000	9	12	BC	USA
Colorado	32000	8	16	BB	USA
Fuso	35000	12	14	BB	Japan
Hood	41000	8	15	BC	Gt. Br.
Iowa	46000	9	16	BB	USA
Ise	36000	12	14	BB	Japan
King George V	38000	10	14	BB	Gt. Br.
Kongo	32000	8	14	BC	Japan
Nagato	38000	8	16	BB	Japan
Nelson	34000	9	16	BB	Gt. Br.
Nevada	29000	10	14	BB	USA
New Mexico	33000	12	14	BB	USA
New York	27000	10	14	BB	USA
North Carolina	37000	9	16	BB	USA
Pennsylvania	33000	12	14	BB	USA
Queen Elizabeth	31000	8	15	BB	Gt. Br.
Renown	32000	6	15	BC	Gt. Br.
Revenge	29000	8	15	BB	Gt. Br.
South Dakota	37000	9	16	BB	USA
Tennessee	32000	12	14	BB	USA
Wyoming	26000	12	12	BB	USA
Yamato	65000	9	18	BB	Japan

Fig. 1. The relation *Classes* of capital ships of the major WW-II naval powers.

Example: In Fig. 1 is a table of capital ship classes.

- The six columns correspond to attributes class name, tonnage, number of guns, caliber of guns, type, and country.
- For example, the first row says that ships of the Alaska class displaced 28,000 tons, mounted nine 12-inch guns, was a class of battle cruisers (somewhat more lightly armored

than the battleships denoted BB), and belonged to the USA.

Why Relations?

- Database industry uses relations as the most important model for representing data.

- The set operations form the basis for database query languages such as SQL that are used to manipulate these databases.

- The supplanting of earlier, ad-hoc database systems by systems based on formal models and ideas is probably the greatest single example of how theory leads to improved practice and ultimately to commercial success.

Dictionary Operations on General Relations

We may describe subsets of tuples by *selection conditions* indicating for each column either

1. A specific value the tuples must have, or

2. No restriction, indicated by a *.

- The generalized *lookup* operator takes a selection condition C and a relation R to which it is applied. It returns the relation of all those tuples in R that match C.

Example: Let C be the selection condition (*,*,10,14,*,*). Then *lookup(C,Classes)* asks for those classes of ships with 10 14-inch guns. It produces:

Class	Weight	Guns	Caliber	Type	Country
King George V	38000	10	14	BB	Gt. Br.
Nevada	29000	10	14	BB	USA
New York	27000	10	14	BB	USA

- Deletion also uses a selection condition and a relation, from which it deletes all the tuples matching the condition.

Example: *delete(C,Classes)* would delete the above 3 tuples from relation Classes.

- Insertion only makes sense if all components of the tuple are specified; i.e., no *'s.

Example: *insert((Vittorio Veneto, 41000, 9, 15, BB, Italy), Classes)* adds to relation Classes the single tuple indicated.

Efficiency of Access

The data structure used to store a relation influences greatly the time it takes to perform a lookup or other operation.

Example: Consider relation Course-Student, whose pairs are a course and a student taking that course.

- If we build a hash table with domain = Course, we get efficiency for conditions like (CS109A, *).
- But no help for a condition like (*, Alan Hu).
  - □ We must search all the buckets.

Keys

An attribute (or set of attributes) is a *key* for relation R if we do not expect to find more than one tuple of R that agree in the key attribute(s).

- Important: keyness depends on our notion of what might be in R at some time.
  - □ It is insufficient that no tuples agree in a set of attributes today, if the relation could legitimately have two agreeing tuples sometime.

Example: For Classes, we expect that Class is a key, since there are unlikely to be two classes with the same name.

- However, two classes from different countries *might* bear the same name, so perhaps {Class, Country} is a safer choice of key.

Why Keys?

- Useful choice as the domain in a data structure.

- E.g., a hash table with a key as domain need not send many tuples to the same bucket for having the same domain value.
- In contrast, Course-Student has a nonkey domain and must hash all pairs with a course like CS109A to the same bucket.