Databases Supervision Work 1
http://www.matthew.ath.cx/notes/

Matthew Johnson
Trinity Hall
mjj29-notes@srcf.ucam.org

May 14, 2003

E/R Modelling

See attached E/R diagram. Note that the dashed underlined attributes indicate foreign keys in other tables - for 1:1 relationships. Multi-way relationships indicated by relationship tables.

y2002p6q8

a) Define the core Relational Algebra

The core relational algebra is made up from:

The *selection* operator σ, which applies a condition C to a relation R and only returns the tuples in R that satisfy C. Written $\sigma_C(R)$, the result has the same schema as R.

The *projection* operator π, which takes as its argument a field list, and it returns all the tuples in R, but only the fields in the field list. Written $\pi_{A_1,\ldots,A_k}(R)$, the resultant schema is a subset of that of R, but it can only remove fields, not rename or change the domain of them.

The *rename* operator ρ, which changes the name of some of the fields in R. The result schema is the same as for R, but with some of the fields renamed.

The standard set theoretic operators ($\cup, \cap$, etc) can all be applied, but in general the schemas of the sets you are operating on must be identical.

The *cartesian product* operator $\times$, which forms the product of all of the rows in A with all of the rows in B. The result of $A \times B$ has a schema which is the sum of that of A and B.

b) Extensions to Relational Algebra

(i) Full Outer Join

The *Full Outer Join* operator $^\circ\bowtie^\circ$ is a composition of the basic relations. A join ($\bowtie$) is defined as the cartesian product of two relations, followed by a selection on the condition of matching shared field. So in the case of relations A and B sharing a field f, $A \bowtie B \equiv \sigma_{A.f=B.f}(A \times B)$. The full outer join operator doesn't remove the tuples for which there isn't a match in the other relation, it just fills the spare fields with NULLs.

(ii) Aggregation and Grouping

There are several operations that you can apply to columns of a relation as a whole. The aggregation operators (SUM, AVG, MIN, MAX COUNT) can be applied to a select query. The result is then a single tuple containing a value that is the result of the operator applied to the whole column. For example, the query `SELECT AVG(s.age) FROM sailors s` would return the average age of all of the sailors.

You can also apply the aggregation operators to subsets of the rows, using the `GROUP BY` syntax. This specifies another field. The aggregation is then calculated over tuples with the same value of the group by field.

This is represented in relational algebra by the γ operator for grouping, and you can specify as its argument an aggregation operator to perform over its group.

c)

Query: `SELECT x.a FROM x,y,z WHERE x.a=y.a OR x.a=z.a`

(i)

The query would be represented in relational algebra as $\pi_{x.a}((x \bowtie y) \cup (x \bowtie z))$.

(ii)

The required set theory expression was $x \cap (y \cup z)$. In set theory this can be expanded to $(x \cap y) \cup (x \cap z)$. This would be equivelent to the given query - as we saw from the relational algebra, assuming thatthere are no duplicates. SQL operates on multisets, rather than sets, and the equivelence I just gave is not valid for multisets. You can insert `DISTINCT` or δ operators to the query or relational algebra, which will perform the same function.