Relational Algebra

And Relational Calculus

Covered To date:

- Practical Databases
  - Data Definition Language
  - Data Manipulation Language
  - Session Control
  - (Also PL/SQL)
- Practical Database Design
  - Entity Relationship Modelling
  - Object-oriented modelling

Remember…

- Relation = a table with columns and rows.
- Attribute = a named column of a relation.
- Domain = the set of allowable values for one or more attributes *
- Tuple = a row of a relation.
- A relation has a name and is made up of named attributes.
- Each tuple contains one value per attribute.

**** Will be covered a little more***

Domain

- Every attribute in a relation is defined on a domain.
- The domain definition can be held centrally, allowing the user to define the meaning and source of values that attributes can hold.
- This avoids semantically incorrect relational operations.
- Attributes from the same domain can be compared, whereas those from different domains cannot.

Degree and Cardinality

- The degree of a relation is the number of ***attributes*** it contains.
- The cardinality of a relation is the number of *tuples* it contains.

Branchno	Street	City	Postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St.	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Cardinality

Degree

Connolly, Begg, 2005

Set

- A definition of a group of objects that contain no duplicates.
- E.G. a business can have a set of suppliers and a set of customers.
  - Each element of the supplier set is unique.
  - Each element of the customer set is unique.
  - There may be a customer who is also a supplier, in which case, this element will belong to both sets.

Relational Database

- A collection of normalised relations with distinct relation names.
  - *If you forget normalisation, go back over it yourselves.*

Theoretical underpinning

- Relational Algebra
  - This is a theoretical language with operations that work on one or more relations to define another relation, without changing the original relation(s).
  - Both the operands and the results are relations.
  - The output from one operation can become an input into another.
  - Operations can be nested.
  - Relations are closed under the algebra, just as numbers are closed under arithmetical operations.

Relational Algebra

- Relational Algebra is the formal description of how a relational database operates.
- It provides the mathematics underpinning SQL operations.
- The OPERATIONS have
  - An operator (i.e. the function)
  - One or more operands.
    - A UNARY operation has only 1 operand.
    - i.e. it works on only 1 relation.

Five fundamental operations

- Selection
- Projection
- Cartesian product
- Union
- Set difference.

Extra operations

- Join
- Intersection
- Division
- These can all be expressed in terms of the five fundamental operations.

Symbols for these operations

- The Greek letter sigma σ represents the ***selection*** operation.
- The Greek letter pi π represents the ***projection*** operation.
- The ***Cartesian Product*** is represented by X.
- The Greek letter upsilon υ represents ***union***.
- The symbol – (minus) represents ***difference***.

Selection

- Also known as Restriction.
- The selection operation works on a single relation R and defines a relation that contains only those tuples of R that satisfy the specified condition (predicate).
  - Is represented as: $\sigma_{predicate}$ **(R)**
- E.g.
  - List all staff with a salary greater than €10,000

    $\sigma_{salary>10000}$ **(staff)**

Selection

- E.g.
  - List all staff with a salary greater than €10,000

    $\sigma_{salary>10000}$ **(staff)**
- In SQL terms, this would be represented as:

```
SELECT * FROM staff WHERE salary > 10000;
```

Projection

- This works on a single relation R and defines a relation that contains a vertical subset of R, extracting the values of specified attributes and eliminating duplicates.

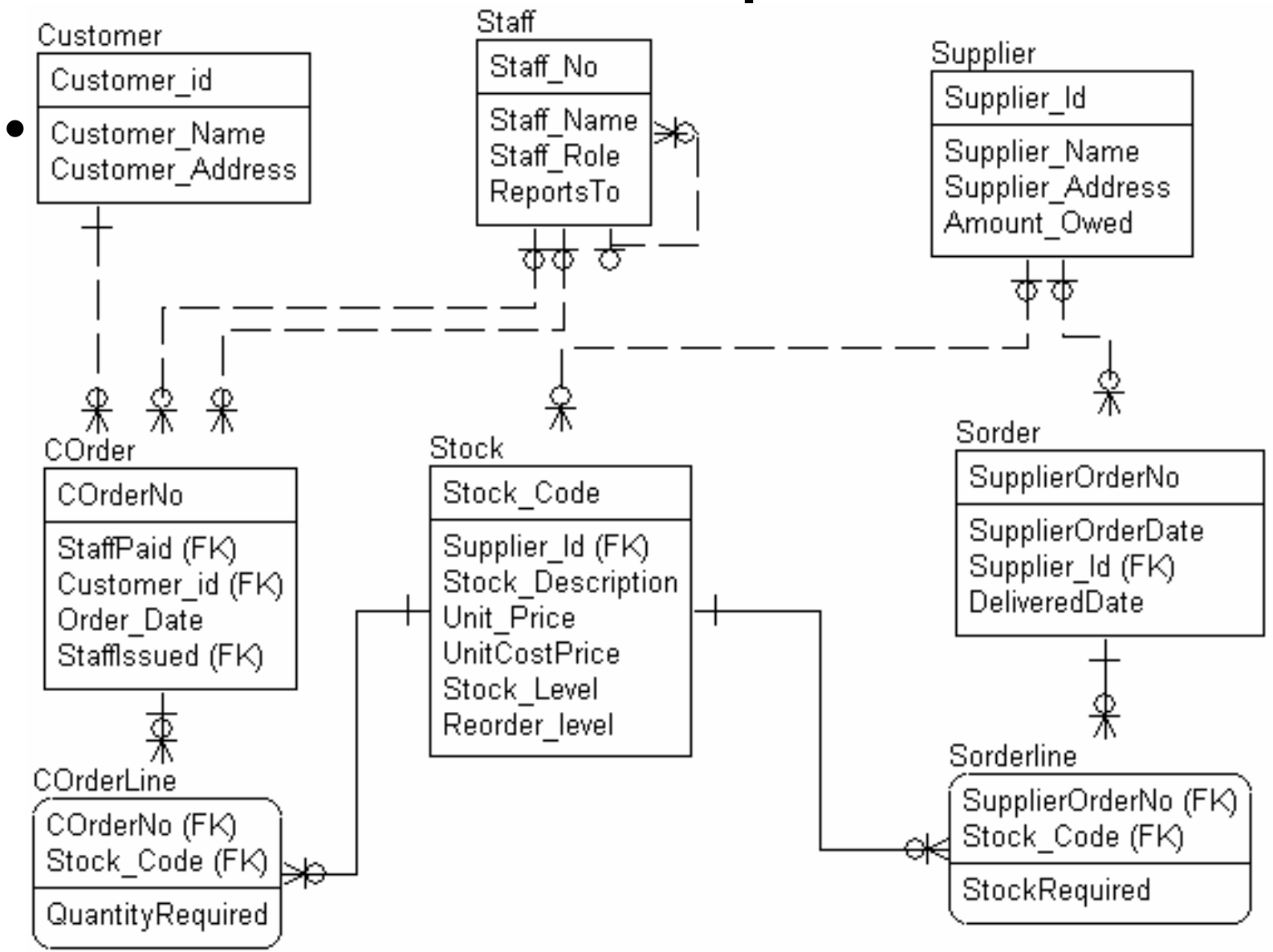
  $\pi_{a_1,a_2,\ldots,a_n}(R)$

- $\pi_{\text{STREET, CITY}}(\text{BRANCH})$
- In SQL terms, this would be represented as:
  - SELECT street, city FROM branch;

Union

- R∪S
  - The union of two relations R and S defines a relation that contains all the tuples of R, *or* S, *or* both R and S, duplicate tuples being eliminated. R and S must be union compatible.
  - Union compatible means that the schemas of the two relations match
    - They have the same number of attributes, with each pair of corresponding attributes having the same domain.
    - Attribute names are not used in defining union compatibility.
    - Sometimes the projection operation is used to make the relations union compatible.

Example


Customer
Customer_id
Customer_Name
Customer_Address
Staff
Staff_No
Staff_Name
Staff_Role
ReportsTo
Supplier
Supplier_Id
Supplier_Name
Supplier_Address
Amount_Owed
COrder
COrderNo
StaffPaid (FK)
Customer_id (FK)
Order_Date
StaffIssued (FK)
Stock
Stock_Code
Supplier_Id (FK)
Stock_Description
Unit_Price
UnitCostPrice
Stock_Level
Reorder_level
Sorder
SupplierOrderNo
SupplierOrderDate
Supplier_Id (FK)
DeliveredDate
COrderLine
COrderNo (FK)
Stock_Code (FK)
QuantityRequired
Sorderline
SupplierOrderNo (FK)
Stock_Code (FK)
StockRequired

Union compatible tables

- Get the names and addresses of all contacts that we use.
- We have supplier names and addresses
  - R1 = $\pi_{supplier\_name,supplier\_address}$ (supplier)
- We have customer names and addresses.
  - R2 = $\pi_{customer\_name,customer\_address}$ (customer)
- We want the union of those:
- R3 = R1uR2 or
- R3 = $\pi_{supplier\_name,supplier\_address}$ (supplier) u $\pi_{customer\_name,customer\_address}$ (customer)

SQL

The SQL version of this is:

Select supplier_name, supplier_address from supplier union select customer_name, customer_address from customer;

Set difference

- R – S
  - The Set difference operation defines a relation consisting of the tuples that are in relation R, but not in S. R and S must be union-compatible.
- Find all suppliers who are not customers:
- R3 = $\pi_{\text{supplier\_name,supplier\_address}}$ (supplier) – $\pi_{\text{customer\_name,customer\_address}}$ (customer)

In SQL

```
SELECT customer_name,
   customer_address
FROM customer
MINUS
SELECT supplier_name, supplier_address
   FROM supplier;
```

Cartesian Product

- This is the last of the five fundamental operations.
- R X S:
  - The cartesian product operation defines a relation that is the concatenation of every tuple of relation R with every tuple of relation S.
    - $R3 = \pi_{supplier\_name}(supplier) \times \pi_{customer\_name,}(customer)$
  - This is the equivalent of
    - SELECT supplier_name, customer_name
    - FROM supplier, customer;

Other operations

- All of these can be expressed in terms of the five fundamentals.
- Intersection
  - $R \cap S$
  - The intersection operation defines a relation consisting of the set of all tuples that are in both R and S.
  - $R \cap S = R - (R - S)$
- Write the relational algebra to return the relation of the names and addresses of all customers who are also suppliers.

The Rename operation

- Uses the Greek letter rho ρ
- $\rho_s(E)$ or $\rho_{s(a_1, a2, \ldots, an)}(E)$