Entity Relationship Model, Relational Data Model, Structured Query Language (SQL)

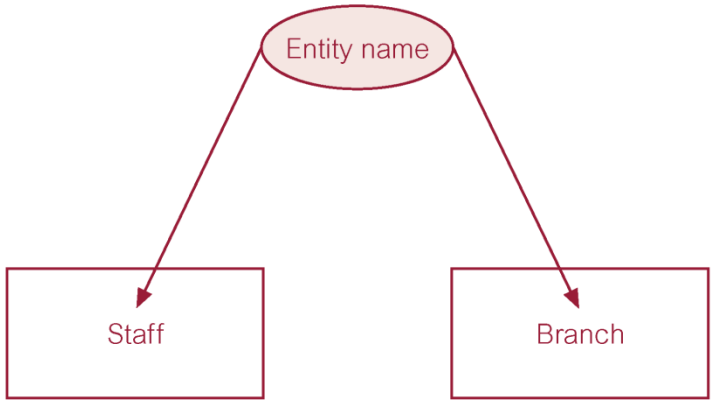
Overview

- Revision of the fundamental concepts
  - Entity Relationship Data Model
    - How to use ER modelling in database design.
    - Basic concepts associated with ER model.
  - Relational Data Model
    - Terminology of relational model.
    - Properties of database relations.
    - Meaning of entity integrity and referential integrity.
    - Etc.
  - Structured Query Language (SQL)
    - Data Manipulation Language (DML) and Data Definition Language (DDL)
- The lecture slides are based on the ones provided with the course textbook:
  - Connolly, T. and Begg, C. "*Database Systems - A Practical Approach to Design, Implementation, and Management*.", 6th Ed., Pearson Education Ltd
- NB: Many slides in the presentation, especially SQL
  - I will quickly go through, or even skip, *some* of them, but will assume you (eventually) understand/know the whole content!

Concepts of ER Model: Entity types

- Entity type
  - Group of objects with same properties, identified by business as having an independent existence.
- Entity occurrence/instance
  - Uniquely identifiable object of an entity type.

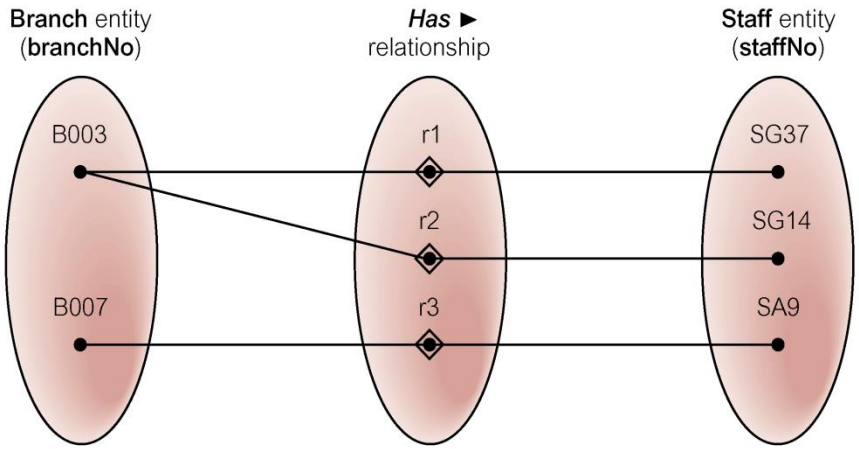
Physical existence	
Staff	Part
Property	Supplier
Customer	Product
Conceptual existence	
Viewing	Sale
Inspection	Work experience



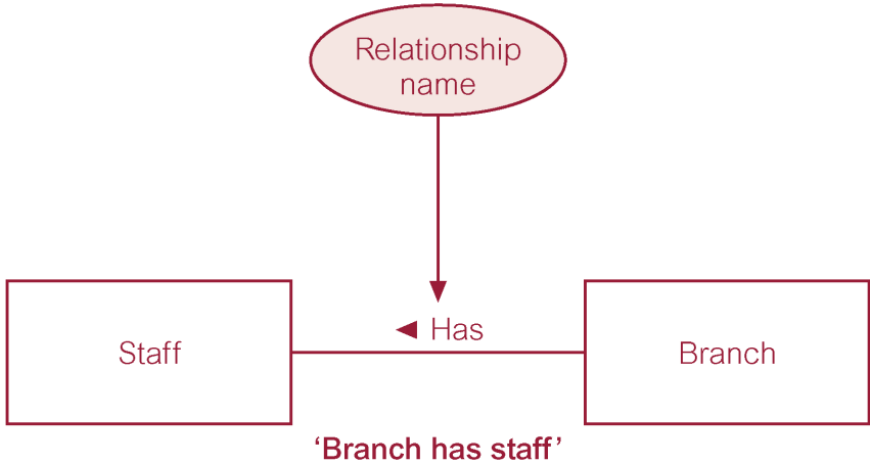
Chen, Peter (March 1976). "The Entity-Relationship Model - Toward a Unified View of Data". ACM Transactions on Database Systems 1 (1): 9–36. doi:10.1145/320434.320440.

Concepts of ER Model: Relationship Types

- Relationship type
  - Set of meaningful associations among entity types.
- Relationship occurrence
  - Uniquely identifiable association, which includes one occurrence from each participating entity type.



Semantic net of *Has* relationship type
They are too detailed – work on level of instances!



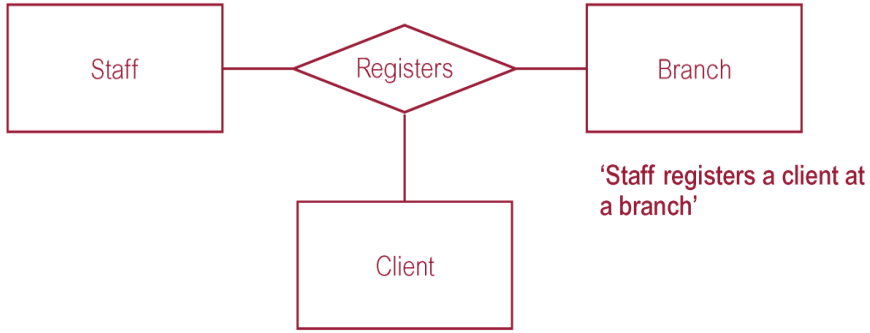
ER diagram of Branch Has Staff relationship

Examples in this lecture from *DreamHouse* case-study in the Connolly and Begg book (see e.g. Sect 11., and especially sub-section 11.4, and/or Appendix A)

Relationship Types

- Degree of a Relationship
  - Number of participating entities
- Relationship degree/arity :
  - two is **binary** (the most common);
  - three is ternary;
  - four is quaternary.
  - Etc.

Binary relationship called *POwns*



Ternary relationship called *Registers*

- Relationships may be given role names to indicate purpose that each participating entity type plays in a relationship.

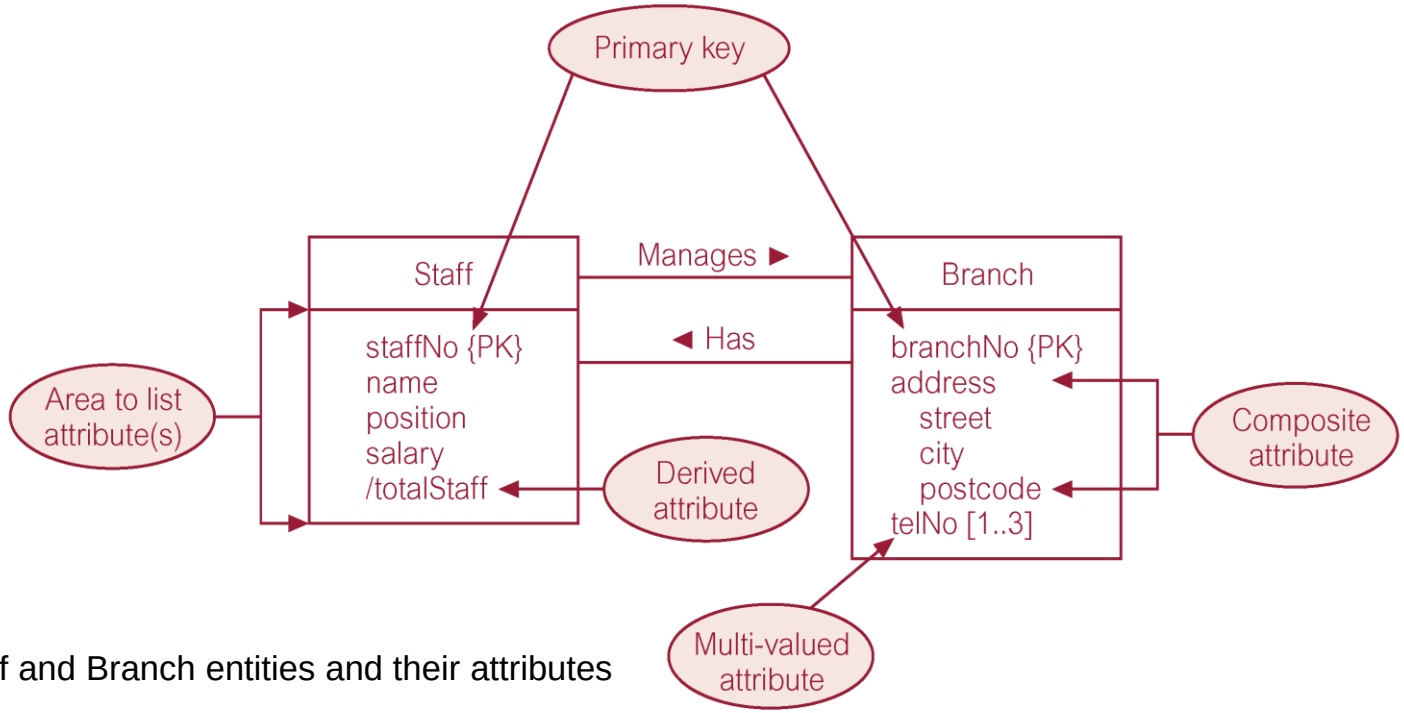
- **Many notations exist!**
- Examples here use UML notation, since many of you have been exposed to it. Other notations are: original Chen notation, Crow's foot notation etc. You need to know them! See e.g. Appendix C of the book .

Concepts of the ER Model: Attributes

- *Attribute*
  - Property of an entity, or a relationship, type.
- Attribute *Domain*
  - Set of allowable values for one or more attributes.
- *Simple* Attribute
  - Attribute composed of a single component with an independent existence.
- *Composite* Attribute
  - Attribute composed of multiple components, each with an indep. existence.
- *Single-valued* Attribute
  - Attribute that holds a single value for each occurrence of an entity type.
- *Multi-valued* Attribute
  - Attribute that holds multiple values for each occurrence of an entity type.
- *Derived* Attribute
  - Attribute that represents a value that is derivable from value of a related attribute, or set of attributes, not necessarily in the same entity type.

Keys

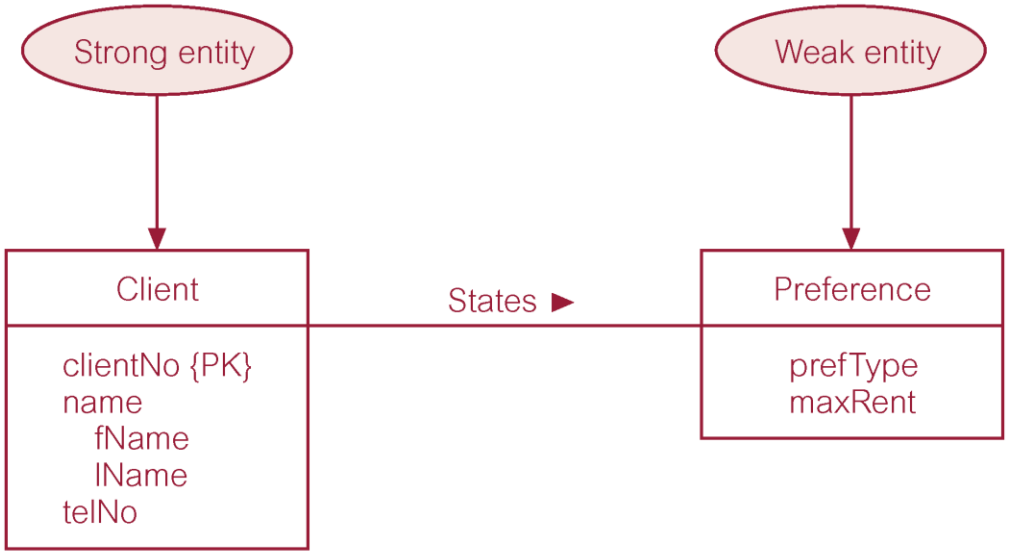
- *Candidate* Key
  - Minimal set of attributes that uniquely identifies each occurrence of an entity type.
- *Primary* Key
  - Candidate key ***selected*** to uniquely identify each occurrence of an entity type.
- *Composite* Key
  - A candidate key that consists of two or more attributes.



ER diagram of Staff and Branch entities and their attributes

Entity Types

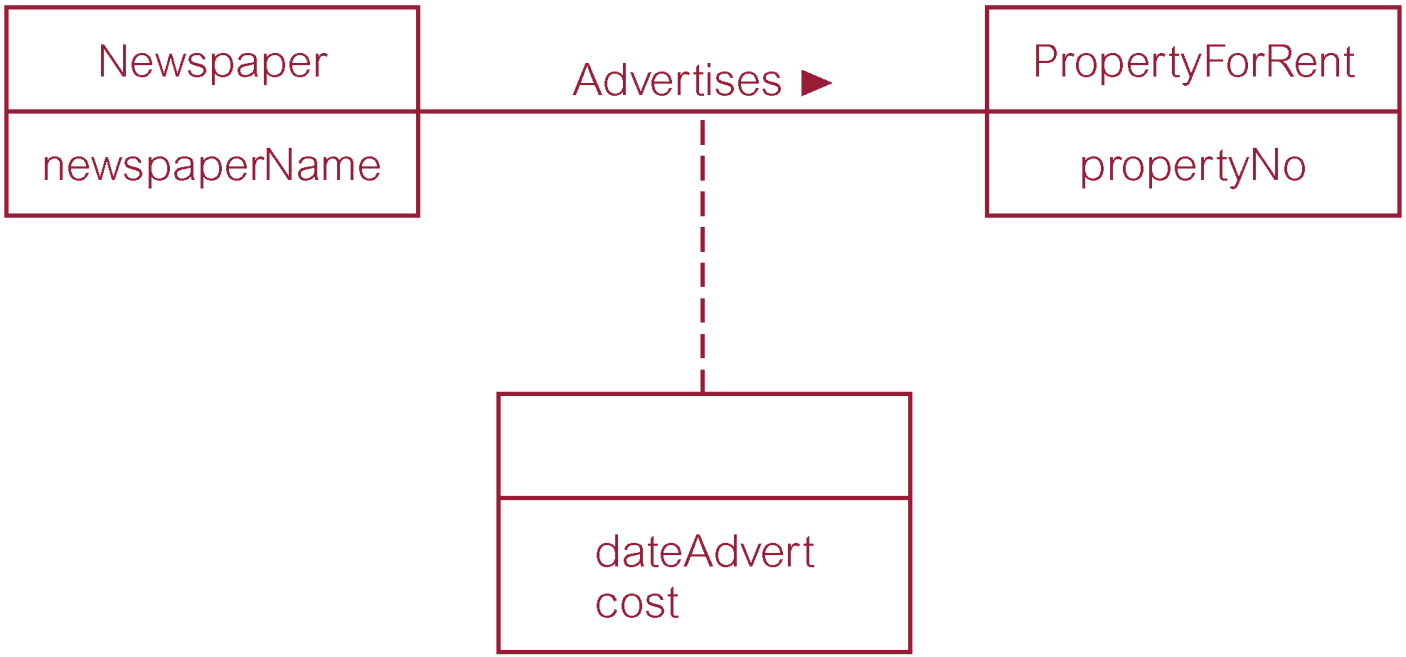
- Strong Entity Type
  - Entity type that is *not* existence-dependent on some other entity type.
- Weak Entity Type
  - Entity type that is existence-dependent on some other entity type.
- *An existential dependency is a constraint requiring that the existence of the instances of one of the entity types in a binary relationship depends on the existence of the other entity type*
- Some ER notations have explicit syntax for this concept



Strong entity type – Client, and weak entity type - Preference

Relationship called *Advertises* with attributes

‘Newspaper advertises property for rent’

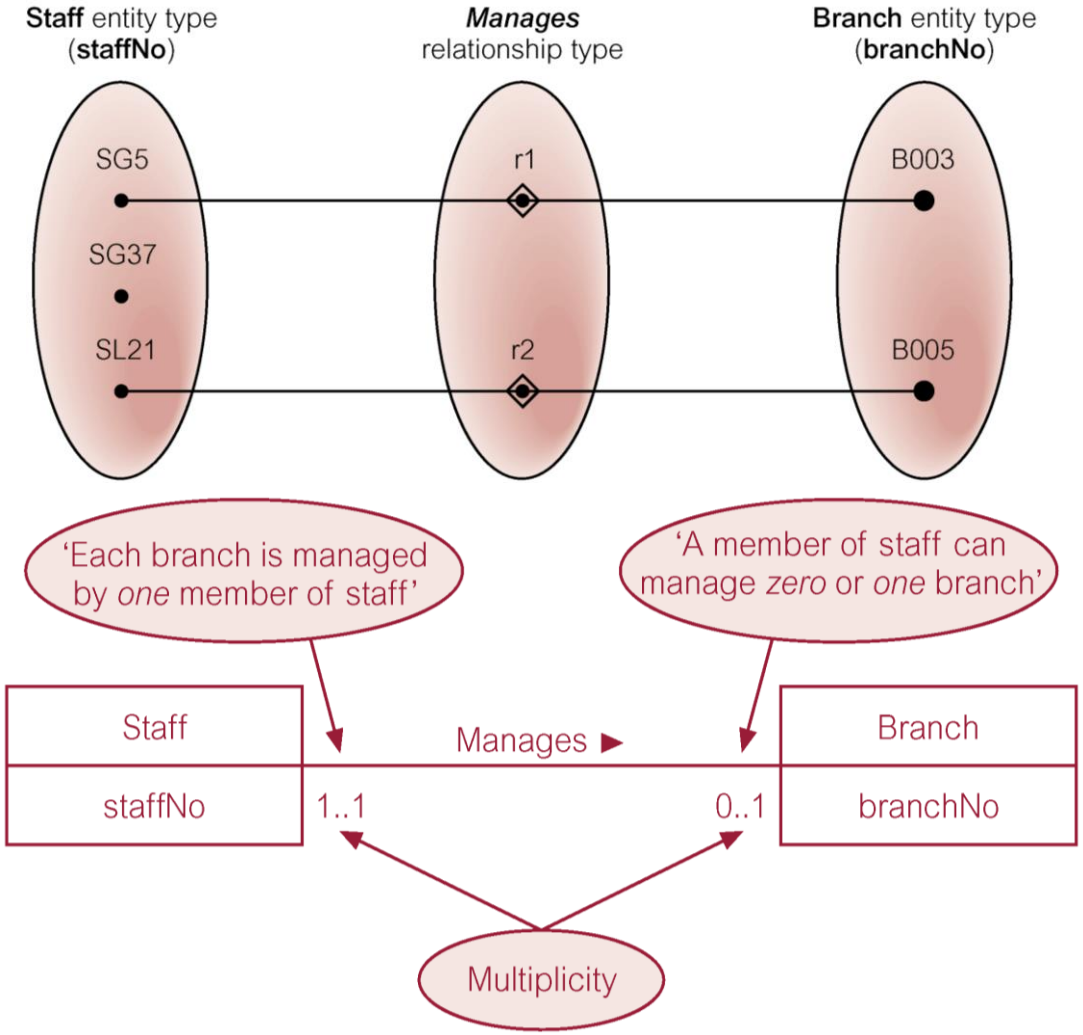


Structural Constraints

- Main type of constraints on relationships is called *multiplicity*.
- *Multiplicity* - number (or range) of possible occurrences of an entity type that may relate to a single occurrence of an associated entity type through a particular relationship.
- Represents policies (called *business rules*) established by user/ stakeholder.

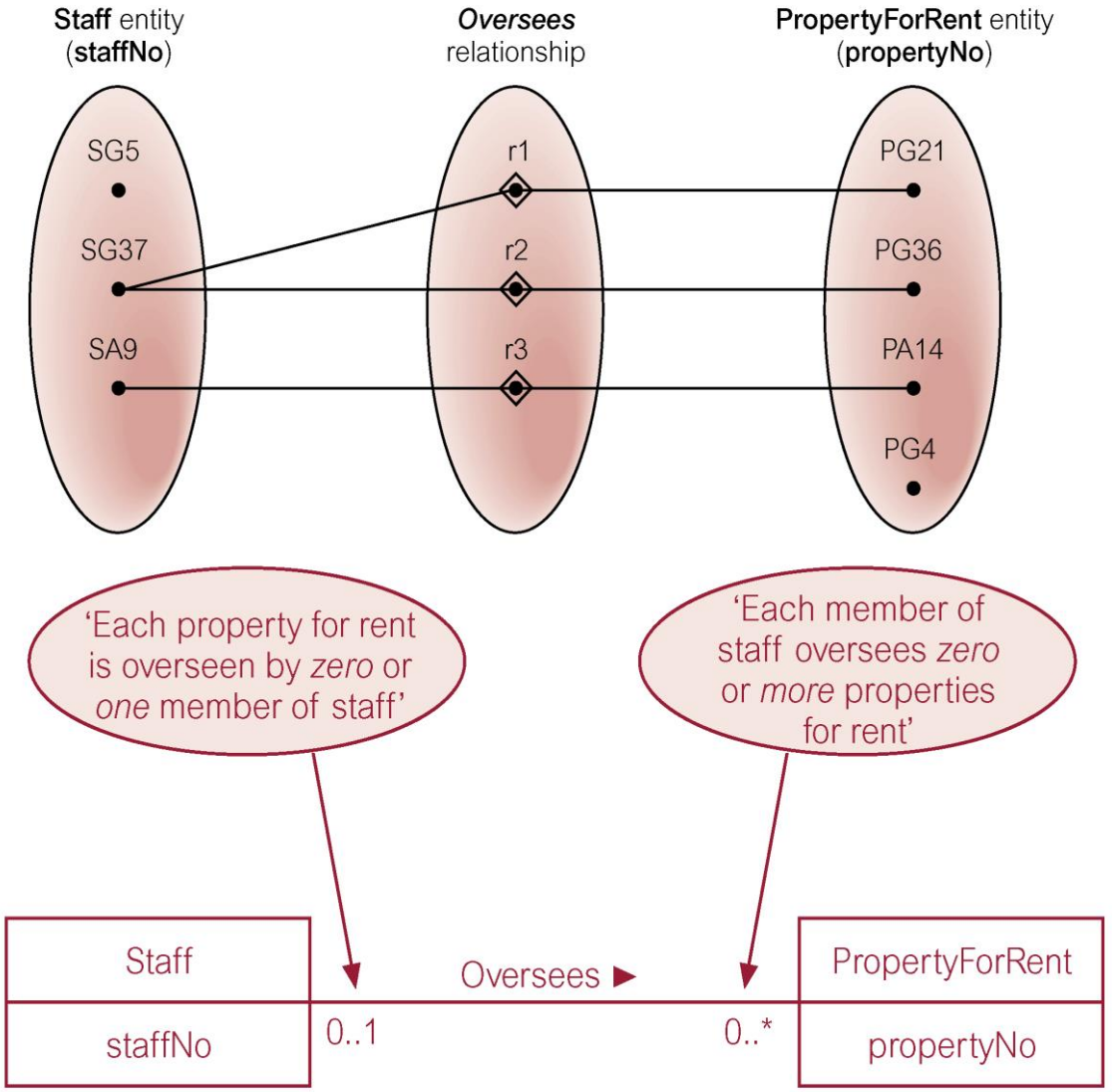
- The most common *degree* for relationships is *binary*.
- Binary relationships are generally referred to as being:
  - one-to-one (1:1); one-to-many (1:*); many-to-many (*:*)

Semantic net, and Multiplicity, of Staff *Manages* Branch relationship type

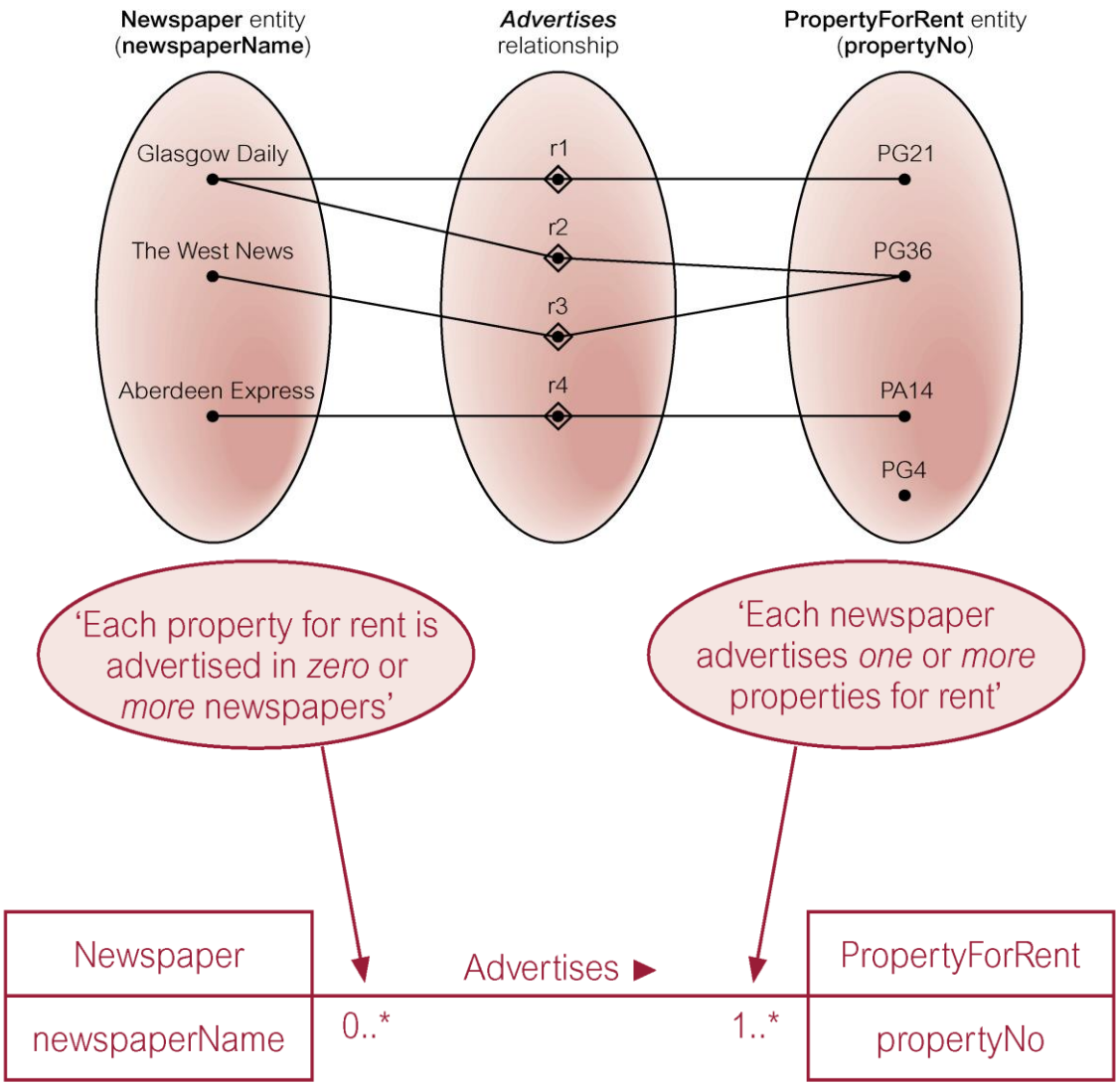


- NB: "Look-here" vs "look-across" multiplicity – depends on the notation; UML-based notation uses the latter.

Semantic net, and Multiplicity, of Staff *Oversees* PropertyForRent relationship type



Semantic net, and Multiplicity, of Newspaper *Advertises* PropertyForRent relationship type



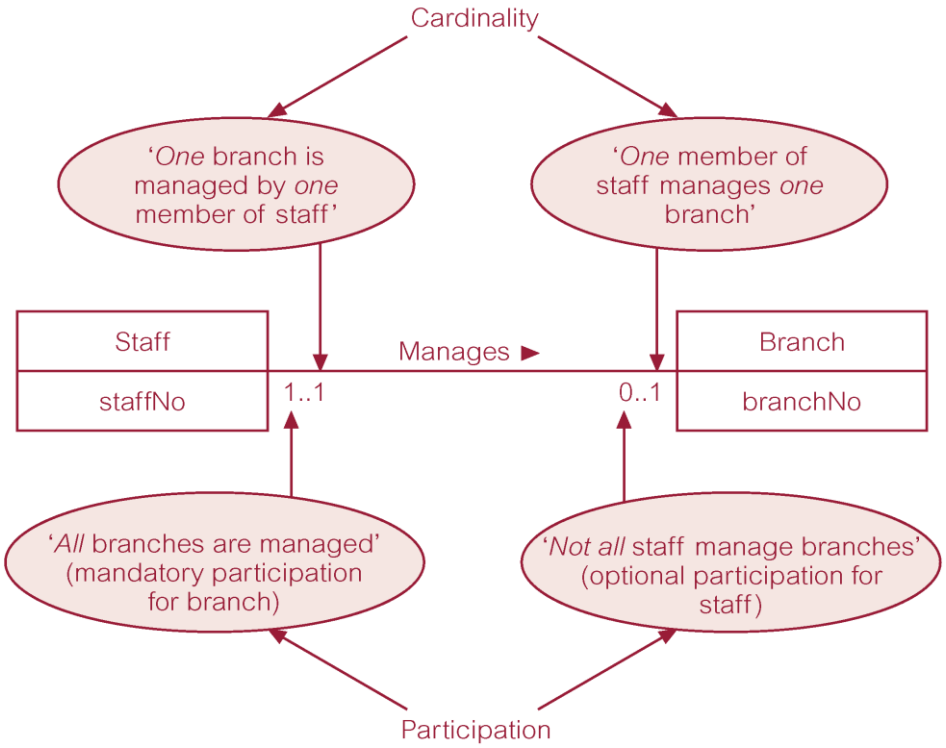
PG4 not advertised in any newspapers – hence zero for PropertyForRent (placed on the Newspaper side – "look across")

Summary of multiplicity constraints

Alternative ways to represent multiplicity constraints	Meaning
0..1	Zero or one entity occurrence
1..1 (or just 1)	Exactly one entity occurrence
0..* (or just *)	Zero or many entity occurrences
1..*	One or many entity occurrences
5..10	Minimum of 5 up to a maximum of 10 entity occurrences
0, 3, 6–8	Zero or three or six, seven, or eight entity occurrences

Structural Constraints

- Multiplicity is made up of two types of restrictions on relationships: *cardinality* and *participation*.
- Cardinality
  - Specifies maximum number of possible relationship occurrences for an entity participating in a given relationship type.
- Participation
  - Determines whether all or only some entity occurrences participate in a relationship.



Enhanced Entity-Relationship (EER) Modelling

- Limitations of basic concepts of the ER model and requirements to represent more complex applications using additional data modelling concepts.
- Since 1980s there has been an increase in emergence of new database applications with more demanding requirements.
- Basic concepts of ER modelling have not been sufficient to represent requirements of newer, more complex applications.
- Response is development of additional 'semantic' modelling concepts.
- Most useful additional data modelling concept of Enhanced ER (EER) model is called *specialization/generalization*.
  - A diagrammatic technique for displaying specialization/generalization in an EER diagram using UML.

- Many ER tools, on City computers too
  - See also https://appsanywhere.city.ac.uk/ , https://labs.city.ac.uk/ .
  - Also, Microsoft Imagine (formerly MS DreamSpark, or even MSDNAA)
    - https://e5.onthehub.com/WebStore/Security/SignIn.aspx?ws=d6c783dc-bf8b-e011-969d-0030487d8897
    - For more info contact Paul Roberts from the IT Specialist Services Team, P.Roberts-1@city.ac.uk

Specialization / Generalization

- Superclass
  - An entity type that includes one or more distinct subgroupings of its occurrences.
- Subclass
  - A distinct subgrouping of occurrences of an entity type.

- Superclass/subclass relationship is one-to-one (1:1).
- Superclass may contain overlapping or distinct subclasses.
- Not all instances of a superclass are instances of a subclass.

- Attribute Inheritance
- An entity in a subclass represents same ‘real world’ object as in superclass, and may possess subclass-specific attributes, in addition to those associated with the superclass.

AllStaff entity holding details of all staff

Specialization/generalization of Staff entity into sub-entities representing job roles

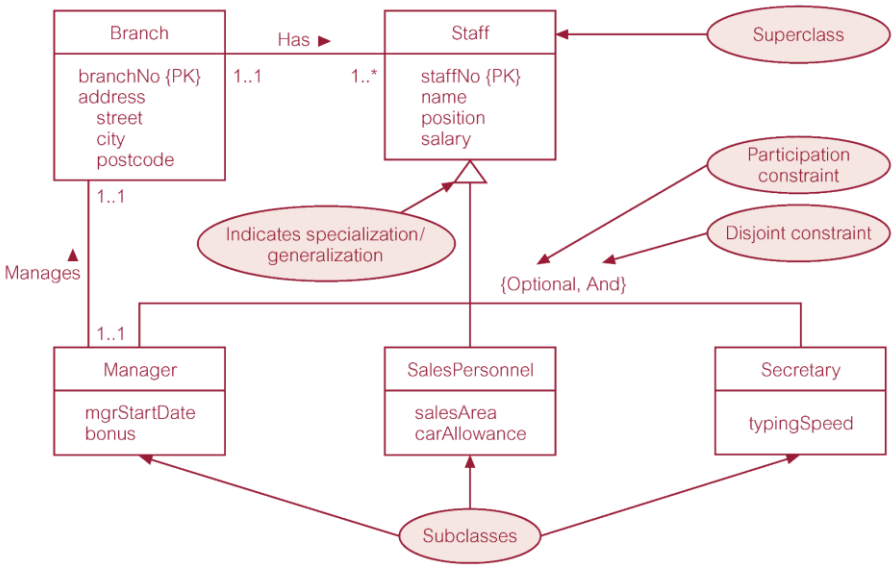
Attributes appropriate for all staff

Attributes appropriate for branch Managers

Attributes appropriate for Sales Personnel

Attribute appropriate for Secretarial staff

staffNo	name	position	salary	mgrStartDate	bonus	sales Area	car Allowance	typing Speed
SL21	John White	Manager	30000	01/02/95	2000			
SG37	Ann Beech	Assistant	12000					
SG66	Mary Martinez	Sales Manager	27000			SA1A	5000	
SA9	Mary Howe	Assistant	9000					
SL89	Stuart Stern	Secretary	8500					100
SL31	Robert Chin	Snr Sales Asst	17000			SA2B	3700	
SG5	Susan Brand	Manager	24000	01/06/91	2350			



- Specialization
  - Process of maximizing differences between members of an entity by identifying their distinguishing characteristics.

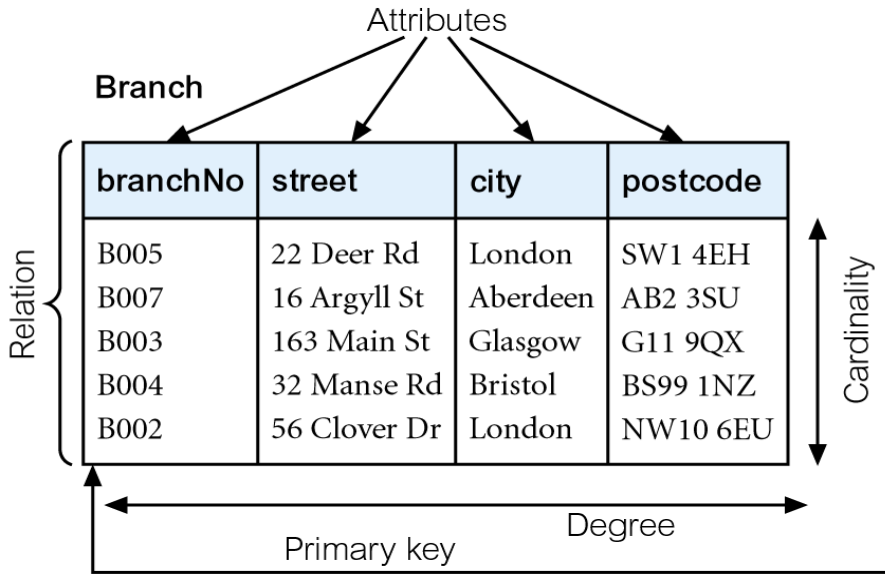
- Generalization
  - Process of minimizing differences between entities by identifying their common characteristics.

The Relational Model

- A *relation* is a table with columns and rows.
  - Only applies to *logical* structure of the database, not the *physical* structure.
- *Attribute* is a named column of a relation.
- *Domain* is the set of allowable values for one or more attributes.

- *Tuple* is a row of a relation.
- *Degree* is the number of attributes in a relation.
- *Cardinality* is the number of tuples in a relation.
- *Relational Database* is a collection of normalized relations with distinct relation names.

Instances of Branch and Staff Relations



Branch

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

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Relation

Examples of Attribute Domains

Attribute	Domain Name	Meaning	Domain Definition
branchNo	BranchNumbers	The set of all possible branch numbers	character: size 4, range B001–B999
street	StreetNames	The set of all street names in Britain	character: size 25
city	CityNames	The set of all city names in Britain	character: size 15
postcode	Postcodes	The set of all postcodes in Britain	character: size 8
sex	Sex	The sex of a person	character: size 1, value M or F
DOB	DatesOfBirth	Possible values of staff birth dates	date, range from 1-Jan-20, format dd-mmm-yy
salary	Salaries	Possible values of staff salaries	monetary: 7 digits, range 6000.00–40000.00

Alternative Terminology for Relational Model

Formal terms	Alternative 1	Alternative 2
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

Database Relations

- Relation schema
  - Named relation defined by a set of attribute and domain name pairs.
- Relational database schema
  - Set of relation schemas, each with a distinct name.
- Relation name is *distinct* from all other relation names in a given relational schema.
- Each cell of relation contains exactly one *atomic* (single) value.
- Each attribute has a distinct name.
- Values of an attribute are all from the same domain.
- Each tuple is distinct; there are no duplicate tuples.
- Order of attributes has no significance.
- Order of tuples has no significance, *theoretically*.
  - Theoretically: in practice the efficiency of accessing rows depends on the order!

Relational Keys

- Superkey
  - An attribute, or set of attributes, that uniquely identifies a tuple within a relation.
- Candidate Key
  - Superkey (K) such that no *proper subset* is a superkey within the relation.
  - In each tuple of R, value(s) of K uniquely identify that tuple (*uniqueness*).
  - No *proper subset* of K has the uniqueness property (*irreducibility*).

- Primary Key
  - Candidate key selected to uniquely identify tuples within relation.
- Alternate Keys
  - Candidate keys that are not selected to be primary key.
- Foreign Key
  - Attribute, or set of attributes, within one relation that matches candidate key of some (possibly same) relation.

Integrity Constraints

- Null
  - Represents value for an attribute that is currently unknown or not applicable for tuple.
  - Deals with incomplete or exceptional data.
  - Represents the absence of a value
    - Not the same as zero or spaces, which are values.

DEPARTMENT TABLE (parent)

DEPTNO	DEPTNAME	DEPTLOC
10	SALES	PITTSBURGH
15	ORACLE	CHARLOTTE

Primary Key

Foreign Key

EMPLOYEE TABLE (child)

EMPNO	ENAME	JOBNO	MGRNO	HIREDATE	DEPTNO
7388	FOOT	87	6599	12-DEC-2015	10
7499	ALLEN	87	6599	06-APR-2010	10
7566	OSTER	115	7521	06-MAY-2011	15

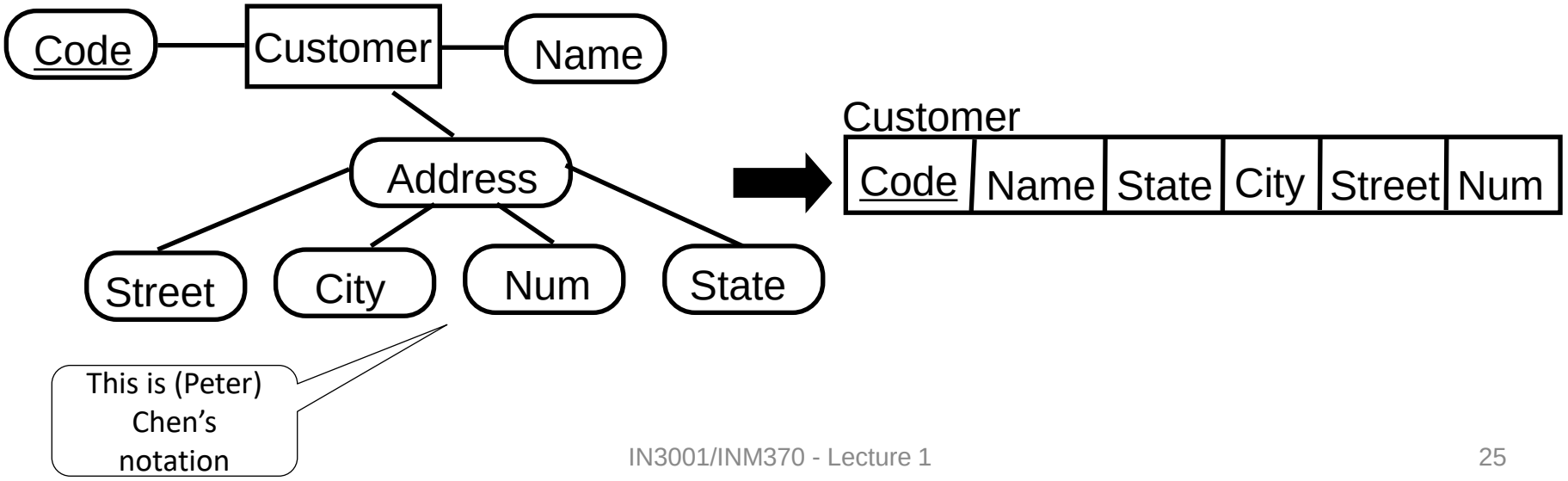
- Entity Integrity
  - In a *base* relation, no attribute of the PK can be null.
- Referential Integrity
  - If a foreign key (FK) exists in a relation, either the FK's value must match a candidate key value of some tuple in its *home* relation, or the FK value must be null.
- General Constraints
  - Additional rules specified by users or database administrators that define or constrain some aspect of the modelled problem domain.

Views

- Base Relation
  - Named relation corresponding to an entity in conceptual schema, whose tuples are physically stored in database.
- View
  - Dynamic result of one or more relational operations operating on base relations to produce another relation.
- A virtual relation that does not necessarily exist in the database but is produced upon request, at the time of request.
  - There exists *materialised view*, however.
- Contents of a view are defined as a query on one or more base relations.
- Views are dynamic, meaning that changes made to base relations that affect view attributes are immediately reflected in the view.

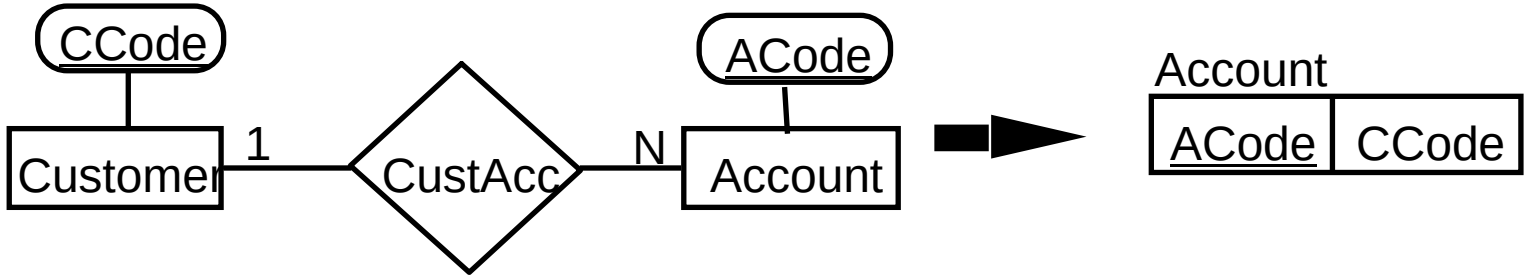
From ER diagrams to Relational Schemas

- The entity types become relations
- The atomic attributes of the entity types become attributes of the relations which represent these types
- The composite attributes of the entity types are not transformed, only their atomic constituents are
- The keys of entity types become the keys of the relations which represent these types

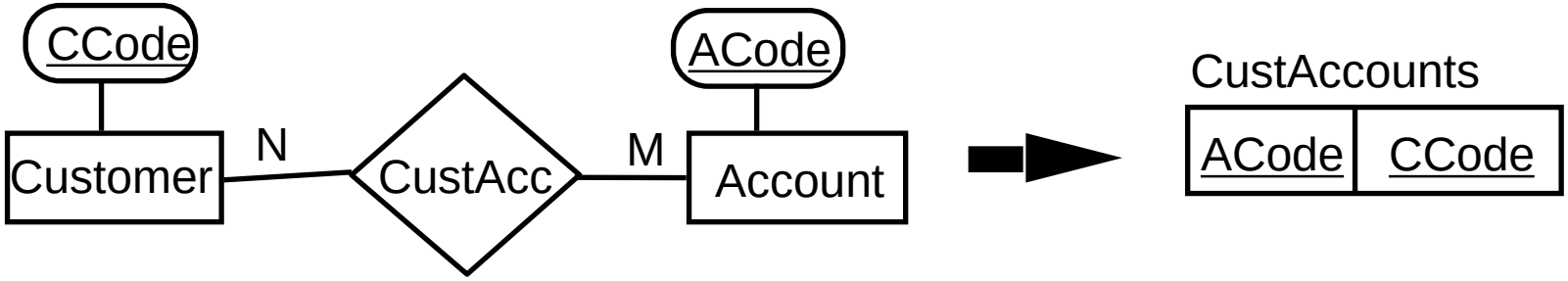


Transformation of Relationship Types

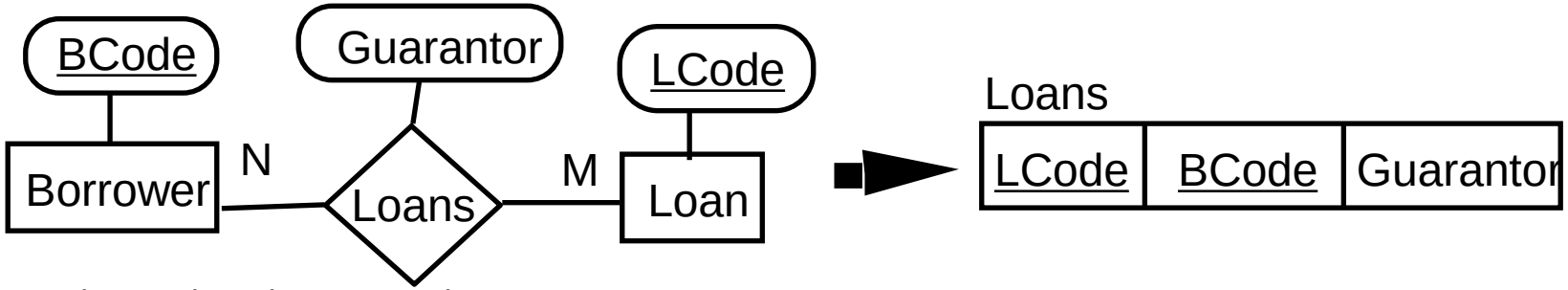
1:N and 1:1 relationships become attributes of relations



N:M relationships become relations



relationships with attributes become relations



Also, n-ary relationships become relations

Structured Query Language (SQL)

- SQL is a (originally) declarative data definition and manipulation language for relational databases based on the relational algebra

- SQL emerged as a descendant of Sequel (*System R*, IBM, 1970)

- standardisation efforts:
- SQL1 standard (ANSI-ISO, 1986)
- SAA-SQL standard (IBM)
- SQL2 standard (ANSI-ISO, 1992)
- on-going standardisation (SQL3, SQL 2003, SQL 2008, SQL 2011, SQL 2016, SQL 2019, ...)

- many commercial variations – **SQL dialects**
  - SQL is the de facto standard for (commercial) relational DBMSs

Main parts of the language

- data manipulation language (DML)
  - *select, delete, insert and update data in relations*
- data definition language (DDL)
  - *create, delete and modify relations*
  - *create indexes*
- data control language (DCL)
  - *grant, revoke*
- integrity constraint specification sublanguage
  - [CONSTRAINT constraint_name] REFERENCES Referenced_Table_name(column_name)
- transaction control sublanguage
  - *start transaction, set isolation level*

SELECT Statement

```
SELECT [DISTINCT | ALL]
    {* | [columnExpression [AS newName]] [,...] }
FROM        TableName [alias] [, ...]
[WHERE      condition]
[GROUP BY   columnList]  [HAVING condition]
[ORDER BY   columnList]
```

FROM Specifies table(s) to be used.

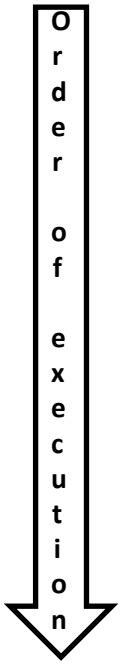
WHERE Filters rows.

GROUP BY Forms groups of rows with same column value.

HAVING Filters groups subject to some condition.

SELECT Specifies which columns are to appear in output.

ORDER BY Specifies the order of the output.



- Order of the clauses cannot be changed.
- Only SELECT and FROM are mandatory.

SELECT; All Columns, All Rows

List full details of all staff.

```
SELECT staffNo, fName, lName, address,
                position, sex, DOB, salary, branchNo
FROM Staff;
```

- Can use * as an abbreviation for 'all columns':

```
SELECT *
FROM Staff;
```

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

SELECT; Use of DISTINCT

- List the property numbers of all properties that have been viewed.
  SELECT propertyNo FROM Viewing;

- Use DISTINCT to eliminate duplicates:
  SELECT DISTINCT(propertyNo) FROM Viewing;

propertyNo
PA14
PG4
PG4
PA14
PG36

propertyNo
PA14
PG4
PG36

(Compound) Comparison Search Condition

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

```
SELECT staffNo, fName, lName, position, salary
 FROM Staff
 WHERE salary > 10000;
```

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG37	Ann	Beech	Assistant	12000.00
SG14	David	Ford	Supervisor	18000.00
SG5	Susan	Brand	Manager	24000.00

- List addresses of all branch offices in London or Glasgow.

```
SELECT *
FROM Branch
WHERE city = 'London'
      OR city = 'Glasgow';
```

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B003	163 Main St	Glasgow	G11 9QX
B002	56 Clover Dr	London	NW10 6EU

Pattern Matching

- Find all owners with the string 'Glasgow' in their address.

```
SELECT ownerNo, fName, lName, address, telNo
FROM PrivateOwner
WHERE address LIKE '%Glasgow%';
```

- SQL has two special pattern matching symbols:
  - %: sequence of zero or more characters;
  - _ (*underscore*): any single character.

- LIKE '%Glasgow%' means a sequence of characters of any length containing the string '*Glasgow*'.

ownerNo	fName	lName	address	telNo
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

NULL Search Condition

- List details of all viewings of property PG4 where a comment has not been supplied.
- There are 2 viewings for property PG4: one with, and one without, a comment.
- Have to test for null explicitly using special keyword IS NULL:

```
SELECT clientNo, viewDate
FROM Viewing
WHERE propertyNo = 'PG4' AND comment IS NULL;
```

- Negated version (IS NOT NULL) checks for non-null values.

Single/Multiple Column Ordering

- List salaries for all staff, arranged in descending order of salary.

```
SELECT staffNo, fName, lName, salary
FROM Staff
ORDER BY salary DESC;
```

staffNo	fName	lName	salary
SL21	John	White	30000.00
SG5	Susan	Brand	24000.00
SG14	David	Ford	18000.00
SG37	Ann	Beech	12000.00
SA9	Mary	Howe	9000.00
SL41	Julie	Lee	9000.00

- To additionally arrange in order of rent, specify *minor* order:

```
SELECT propertyNo, type, rooms, rent
FROM PropertyForRent
ORDER BY type, rent DESC;
```

propertyNo	type	rooms	rent
PG16	Flat	4	450
PL94	Flat	4	400
PG36	Flat	3	375
PG4	Flat	3	350
PA14	House	6	650
PG21	House	5	600

- If *rent* was not in the ORDER BY list, the system would arrange the rows with value “flat” – there are 4 such rows - in any order it chooses (similarly for the 2 rows with the value “house”).

SELECT Statement - Aggregates

- ISO standard defines five aggregate functions:

COUNT returns number of values in the specified column.

SUM returns sum of values in the specified column.

AVG returns average of values in the specified column.

MIN returns smallest value in the specified column.

MAX returns largest value in the specified column.

- Each operates on a single column of a table and *returns a single value*.
- COUNT, MIN, and MAX apply to numeric and non-numeric fields, but SUM and AVG may be used on numeric fields only.
- Apart from COUNT(*), each function eliminates NULLs first and operates only on remaining non-null values.
  - COUNT(*) function counts all rows of a table, regardless of NULLs or duplicates

Use of COUNT(*)

- How many properties cost more than £350 per month to rent?

```
SELECT COUNT(*) AS myCount
FROM PropertyForRent
WHERE rent > 350;
```

myCount
5

- Aggregate functions can be used only in SELECT list and in HAVING clause.

SELECT Statement - Grouping

- Use GROUP BY clause to get sub-totals.
- SELECT and GROUP BY closely integrated: each item in SELECT list must be *single-valued per group*, and SELECT clause may only contain:
  - column names; aggregate functions; constants; expression involving combinations of these.
  - Find number of staff in each branch and their total salaries.

```
SELECT    branchNo,
          COUNT(staffNo) AS myCount,
          SUM(salary) AS mySum
FROM Staff
GROUP BY branchNo
ORDER BY branchNo;
```

branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00
B007	1	9000.00

- All column names in SELECT list must appear in GROUP BY clause unless name is used only in an aggregate function.
- If WHERE is used with GROUP BY, WHERE is applied first, then groups are formed from remaining rows satisfying predicate.
- Two NULLs are equal in GROUP BY (as per the ISO standard).

Multi-Table Queries

- If result columns come from more than one table must use a JOIN.
  - Note: Instead of JOINS, one can use subqueries.
- To perform join, include more than one table in FROM clause.
- Use comma as separator and typically include WHERE clause to specify join column(s).
- Also possible to use an alias for a table named in FROM clause.
- Alias is separated from table name with a space.
- Alias can be used to qualify column names when there is ambiguity.

Simple Join

- List names of all clients who have viewed a property, and include comment supplied (if any).

```
SELECT c.clientNo, fName, lName, propertyNo, comment
FROM Client c, Viewing v
WHERE c.clientNo = v.clientNo;
```

- Only those rows from both tables that have identical values in the clientNo columns (c.clientNo = v.clientNo) are included in result.

clientNo	fName	lName	propertyNo	comment
CR56	Aline	Stewart	PG36	
CR56	Aline	Stewart	PA14	too small
CR56	Aline	Stewart	PG4	
CR62	Mary	Tregear	PA14	no dining room
CR76	John	Kay	PG4	too remote

Alternative JOIN Constructs

- SQL provides alternative ways to specify joins:

  FROM Client c JOIN Viewing v ON c.clientNo = v.clientNo
  FROM Client JOIN Viewing USING clientNo
  FROM Client NATURAL JOIN Viewing

- In each case, FROM replaces original FROM and WHERE. However, first produces table with two identical clientNo columns.

- Outer (Left, Right, Full), Inner JOINs
  - Outer join operations retain rows that do not satisfy the join condition.

INSERT

INSERT INTO TableName [(columnList)]
VALUES (dataValueList)

- *columnList* is optional; if omitted, SQL assumes a list of **all** columns in their original CREATE TABLE order.
- Any columns omitted must have been declared as NULL when table was created, unless DEFAULT was specified when creating column.
- *dataValueList* must match *columnList* as follows:
  - number of items in each list must be same;
  - must be direct correspondence in position of items in two lists;
  - data type of each item in *dataValueList* must be compatible with data type of corresponding column.
- Insert a new row into Staff table supplying data for all columns.

INSERT INTO Staff VALUES ('SG16', 'Alan', 'Brown', 'Assistant', 'M', '1957-05-25', 8300, 'B003');

Or a subset of columns

INSERT INTO Staff *(staffNo, fName, lName, position, salary, branchNo)* VALUES ('SG44', 'Anne', 'Jones', 'Assistant', 8100, 'B003');

UPDATE

```
UPDATE TableName
SET columnName1 = dataValue1
      [, columnName2 = dataValue2...]
[WHERE searchCondition]
```

- *TableName* can be name of a base table or an updatable view.
- SET clause specifies names of one or more columns that are to be updated.
- WHERE clause is optional:
  - if omitted, named columns are updated for all rows in the table;
  - if specified, only those rows that satisfy *searchCondition* are updated.
- New *dataValue(s)* must be compatible with data type for corresponding column.
- Give all staff a 3% pay increase.
  - UPDATE Staff SET salary = salary*1.03;

DELETE

DELETE FROM TableName
[WHERE searchCondition]

- *TableName* can be name of a base table or an updatable view.
- *searchCondition* is optional.
  - If *search_condition* is specified, only those rows that satisfy condition are deleted.
  - If omitted, all rows are deleted from table.
    - This does not delete the table, however! See DROP TABLE below, instead.
- Delete all viewings that relate to property PG4.
  - DELETE FROM Viewing WHERE propertyNo = ‘PG4’;
- Delete all records/rows from the Viewing table.
  - DELETE FROM Viewing;

ISO SQL Data Types

DATA TYPE	DECLARATIONS				
boolean	BOOLEAN				
character	CHAR	VARCHAR			
bit†	BIT	BIT VARYING			
exact numeric	NUMERIC	DECIMAL	INTEGER	SMALLINT	BIGINT
approximate numeric	FLOAT	REAL	DOUBLE PRECISION		
datetime	DATE	TIME	TIMESTAMP		
interval	INTERVAL				
large objects	CHARACTER LARGE OBJECT		BINARY LARGE OBJECT		

†BIT and BIT VARYING have been removed from the SQL:2003 standard.

Data Definition

- SQL DDL allows database objects such as schemas, domains, tables, views, and indexes to be created and destroyed.
- Main SQL DDL statements are:

CREATE SCHEMA	DROP SCHEMA
CREATE/ALTER DOMAIN	DROP DOMAIN
CREATE/ALTER TABLE	DROP TABLE
CREATE VIEW	DROP VIEW

- Many DBMSs also provide:

CREATE INDEX / DROP INDEX

Data Definition

- Relations and other database objects exist in an *environment*.
- Each environment contains one or more *catalogs*, and each catalog consists of set of schemas.
- Schema is named collection of related database objects.
- Objects in a schema can be tables, views, domains, assertions, collations, translations, and character sets. All have same owner.

CREATE TABLE

- Creates a table with one or more columns of the specified *dataType*.
- When NOT NULL, system rejects to record a null in the column.
- Can specify a DEFAULT value for the column.
- Primary keys should always be specified as NOT NULL.
- FOREIGN KEY clause specifies FK along with the referential action.

```
CREATE TABLE TableName
{(colName dataType [NOT NULL] [UNIQUE]
[DEFAULT defaultOption]
[CHECK searchCondition] [,...]}
[PRIMARY KEY (listOfColumns),]
{[UNIQUE (listOfColumns),] […,]}
{[FOREIGN KEY (listOfFKColumns)
  REFERENCES ParentTableName [(listOfCKColumns)],
  [ON UPDATE referentialAction]
  [ON DELETE referentialAction ]] [,…]}
 {[CHECK (searchCondition)] [,…] })
```

ALTER TABLE

- Set a default value for a column.
- Drop a default value for a column.
- Add a new column to a table.
- Drop a column from a table.
- Add a new table constraint.
- Drop a table constraint.

- Change Staff table by removing default for position column and setting default for gender column to female ('F').

  ALTER TABLE Staff ALTER position DROP DEFAULT;

  ALTER TABLE Staff ALTER gender SET DEFAULT 'F';

DROP TABLE

DROP TABLE TableName [RESTRICT |CASCADE]

e.g. DROP TABLE PropertyForRent;

- Removes named table and all rows within it.
- With RESTRICT: if any other objects depend for their existence on continued existence of this table, SQL does not allow request.
- With CASCADE: SQL drops all dependent objects (and objects dependent on these objects).

Views

View

Dynamic result of, one or more, relational operations, which operate on base relation(s) to produce another relation.

- Virtual relation that does not necessarily exist in the database but is produced upon request, at the time of request.
- Contents of a view are defined as a query on one or more base relations.
- With view resolution, any operations on view are automatically translated into operations on relations from which it is derived.
- With view materialization, the view is stored as a temporary table, which is maintained as the underlying base tables are updated.

Create View

- Create view of staff details at branch B003 excluding salaries.

```
CREATE VIEW Staff3
AS
    SELECT staffNo, fName, lName, position, sex
    FROM Staff
    WHERE branchNo = 'B003';
```

Then:

```
SELECT * FROM Staff3
```

staffNo	fName	lName	position	sex
SG37	Ann	Beech	Assistant	F
SG14	David	Ford	Supervisor	M
SG5	Susan	Brand	Manager	F

Required Reading

- Connolly, T. and Begg, C. "*Database Systems - A Practical Approach to Design, Implementation, and Management*.", 6th Ed., Pearson Education Ltd
  - Entity Relationship Data Model: Sections 12.1-12.6, 13.1
  - Relational Model: Chapter 4.
  - SQL (DML and DDL): Chapter 6, Sections 7.1-7.4

- This list is only indicative, and the text in these sections should be of course looked at as "additional", but required, reading to the lecture slides.
- This is certainly not an exhaustive list.
- The content of the chapters referred to might NOT be *fully* relevant to the module. You need to make the final decision about what is relevant and what is not yourselves, given the material that we covered in the lecture.