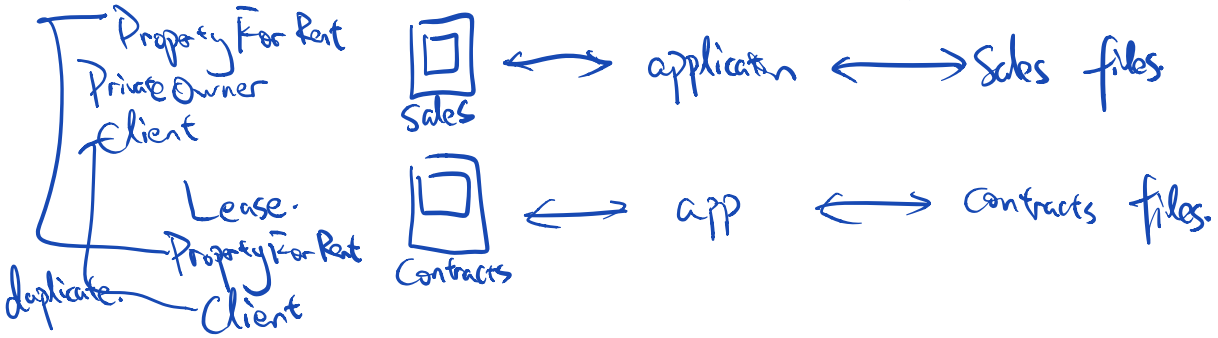
COMP0009 DB.

Concept.

file-based systems.: ! systems that store data are not db systems.

application perform services for users
each defines and manages own data.



limitations

- Separation of data.
  (unaware of data held by other dept.).
- duplication of data.
  - same dat held twice.
  - potential collisions / different formats.
- data dependence. (on program)
  - file structure defined by program. code.

causing ↓

- Incompatible file formats
- fixed queries. (new query → new program)

Database Approach.

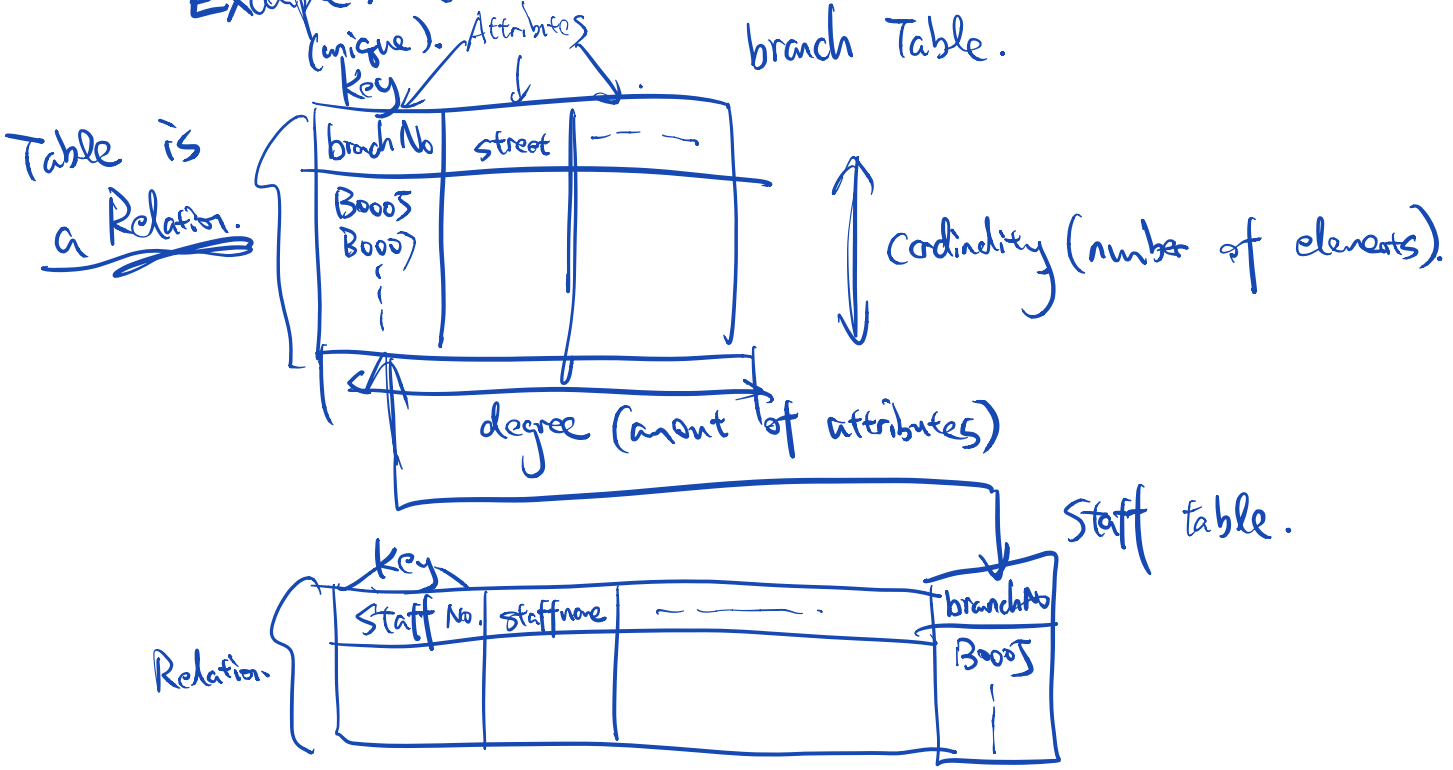
Definition. - a shared collection of logically related data.
- a self-describing collection of integrated records.

Data - entities
attributes (describe entity)
relationships.

(relational data model.)

data catalogue — description data

Example. (relational data model)



rows in table are unique defined by key

History.

first-gen
hierarchical & network.

second-gen
relational DB 1970s.

third-gen.
object relational
object-oriented

DBMS (Management System)

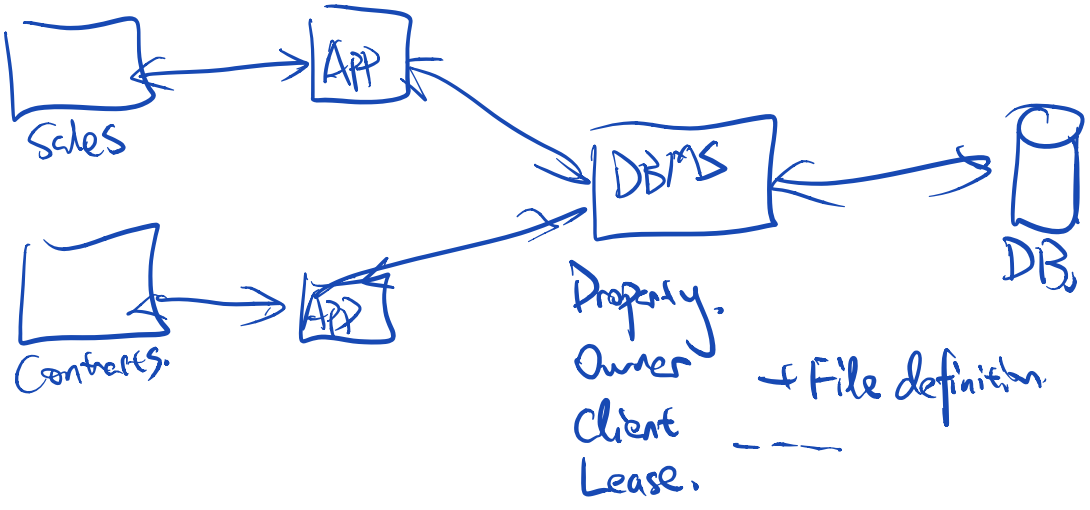
Definition.

system enables users to define create maintain a DB and provides controlled access.

thru a lang. SQL.

provides services around DB.

integrity system.



Primary functions of a DBMS

- a data language Structured Query Lang SQL
- a security system
- a data integrity system.
- a Concurrency Control system
- a recovery system
- a user accessible catalogue. (self-describing)

Advantages of DBMS.

- control of data redundancy
- data consistency
- more information from the same amount of data (from relation)
- + integrity
- + security
- enforcement of data std.
- scale.
- resolve conflicts of requirements
- + concurrency.

- + accessibility & responsiveness

Disadvantages

- Complex
- Size
- conversion
- higher impact of failure.
- cost
- additional hardware.
- performance.

SQL.

a data language. (key tactics of a DBMS)

- create and modify database structures.

  Data Definition Language DDL

- manipulating data content of databases.

  (changing and retrieving actual data)

  Data Manipulation Language DML.

Definition

SQL is a non-procedural and transform-oriented language. (specify what information you require, rather how to get it)

Essentially free form and semi-natural.

ISO Standard.

Create DB.

```
CREATE DATABASE EstateAgent;

USE EstateAgent

CREATE TABLE staff (staffNo VARCHAR(4),
  fName VARCHAR(20), ... salary DECIMAL(5,2),
  ... );

INSERT INTO 'staff' ( 'staffNo', 'fName' ...
  ... 'salary' ... ), VALUES ( 'SG17', 'Joe' ...
  ... 21000, ... )

DELETE FROM Staff WHERE staffNo='SG17';
```

DB can be populated from file.

Update data.

```
UPDATE Staff
SET salary = salary x 1.03.

UPDATE Staff SET salary = salary x 0.95 where
position="Manager"
```

Underlying DB Engine.

InnoDB MyISAM.

↑ newer.

ACID-compliant transaction features

primary key - foreign key relationship support

Standard distribution

Queries.

SELECT	columns to appear.
FROM	table.
WHERE	filter rows
GROUP BY	forms groups of rows each same column val.
HAVING	filter groups subject
ORDER BY	order results

```
SELECT * FROM Staff;

SELECT staffNo, fName, lName, salary FROM Staff;

SELECT DISTINCT propertyNo From Viewing;

SELECT property ---------- (might get duplicates)

SELECT staffNo ---------- salary/12 AS monthlySalary
  From Staff                          only label
```

not Variable

SELECT * FROM STAFF WHERE salary > 10000

Compound comparison search.

SELECT * FROM BRANCH WHERE city = London OR city = Glasgow

SELECT ——— WHERE salary BETWEEN 20000 AND 30000.

or salary ≥ 20000 AND Salary ≤ 30000.

Set membership search. IN

SELECT ——— FROM STAFF WHERE position IN ("Manager", "Supervisor")

Pattern matching: 'LIKE'.

SELECT propertyNo FROM ——— WHERE street LIKE '%ann%'

% .* any char seq.

_ . any char once

LIKE '% Glasgow %'.

LIKE 'G%'.

LIKE 'G _ _ _ _ _'

'IS NULL' search condition.

find stuff where there is no content.

SELECT Details FROM Viewings WHERE comment IS NULL.

IS NOT NULL.

SELECT * FROM Staff ORDER BY salary DESC.

ORDER By type, rent DESC

Aggregate functions.

COUNT SUM AVG MIN MAX.

SELECT COUNT(*) AS count FROM Property WHERE rent > 350

SELECT COUNT (DISTINCT propertyNo) AS Cont.
FROM VIEWING

↑ distinct count of property viewed.

SELECT COUNT (staffNo) AS cont, SUM(salary) AS Sum

FROM staff
WHERE position = "Manager"

Sum of managers' salary

SELECT MIN(salary) AS min FROM staff.

'GROUP BY' grouping sets of rows

SELECT branchNo, COUNT(staffNo) AS cnt,
SUM(salary) AS sum.
FROM staff GROUP BY branchNo.

SELECT may only contain
- column names
- aggregate func.
- constant
- Expr. combined

— — — FROM staff GROUP BY
branchNo HAVING COUNT(staffNo)
ORDER BY branch No

Subsequery

SELECT embedded in a query.

SELECT * FROM staff WHERE branchNo = (
SELECT branchNo
FROM Branch.

WHERE sclary = ...").

SELECT * FROM Staff WHERE sclary >
(SELECT AVG(sclary) FROM Staff)

4 subquery rules.
- ORDER BY should not be used
- Subquery only have one single col. (except exists)
- when used in comparison, should be on right hand side

ANY and ALL.
SOME.

Condition true satisfied by ALL.
by at least one value

SELECT * FROM Staff.
WHERE sclary > SOME(SELECT sclary
FROM Staff
where branchNo
= 'B003').

EXISTS NOT EXIST.
SELECT * FROM Staff S.
WHERE EXISTS (SELECT * FROM Brach b.

WHERE s.branchNo = b.branchNo
AND city = 'London'

Multi-Table Queries

use join if results come from other tables

FROM Staff Join Branch ON Staff.branchNo = Branch.branchNo

Multiple grouping columns GROUP BY. s.branchNo, s.staffNo

Inner Join.

SELECT b.*, p.*
FROM b, p. (excluding unmatched)
WHERE b.city = p.city

Left Outer Join.

SELECT ---- FROM Brach AS b. LEFT JOIN Property AS p ON b.city = b.city
(have unmatched all branches)

Right Outer Join

would have unmatched Property

UNION.

produces result from queries and merge.

AUB (SELECT city from Brach) UNION (SELCT city from Property)

INTERSECT. (no supported by mySQL)

A∩B.

— — INTERSECT — —

EXCEPT

A\B.

DB Design. ER diagram.

— Conceptual. (data modelling with entity relationship)

— Logical DB design.

Lifecycle of DB.

sys. definition

db. planning

requirements gathering

↓

Conceptual design. ERD.

↓

Logical design.

Database schema. (specified)

↓

Normalisation.

Database schema. (verified)

↓

DBMS selection → Physical Design.

DB schema. (optimised.

Conceptual Design.

- understanding semantics of data.
- Communication about info requirements.
- identify variation of user req.

Data model as entity relationship diagram. ERD.
constituted of entities and relationships.

ER

* concepts
  entities, relationships, and attributes

ER diagram
* directed graph.

* entity types / relationship types / attributes

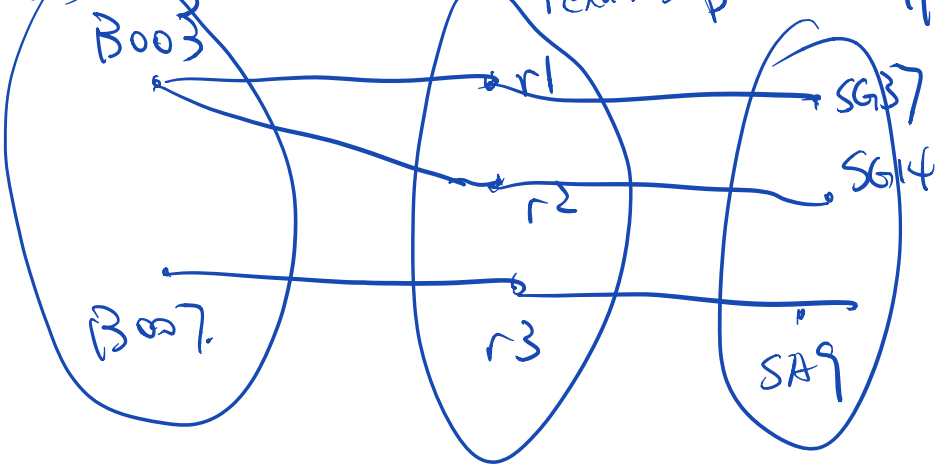
ERD describe types of entity Staff

Entity physical or ~~abstract~~ not a person

Relationship Types

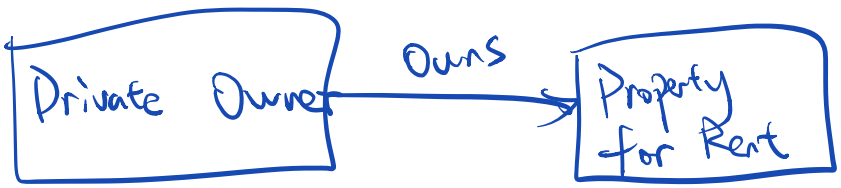
* set of meaningful associations among entity types.

Ex. semantic net diagram
Branch entity relationship Staff entity

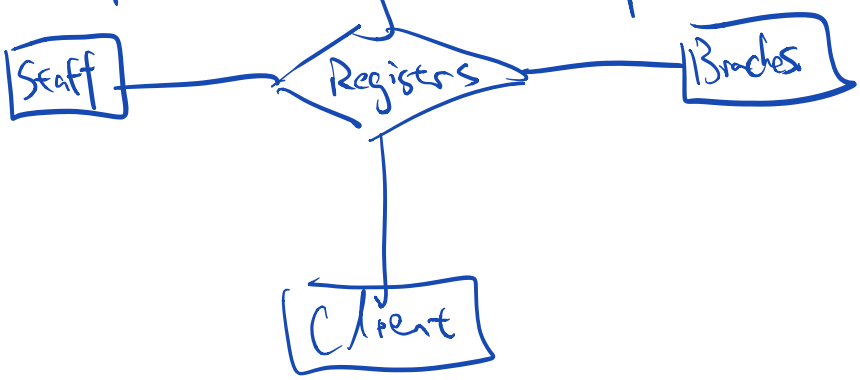


Degree of a relationship. : number of entities involved

binary
ternary
quaternary ---

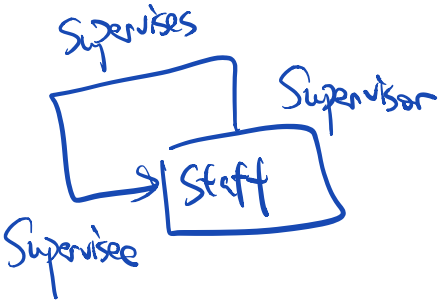


Example of a ternary relationship.

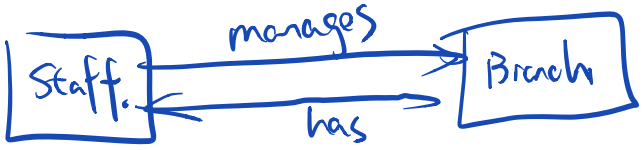


Recursive Relationship.

where same entity type participates more than once in different roles.



Two entities may be associated thru more than 1 relationship



Attributes.

- a property of an entity or a relationship type.
- hold values that describe an entity instance.

Domain
set of allowable values for one or more attributes
Int / String

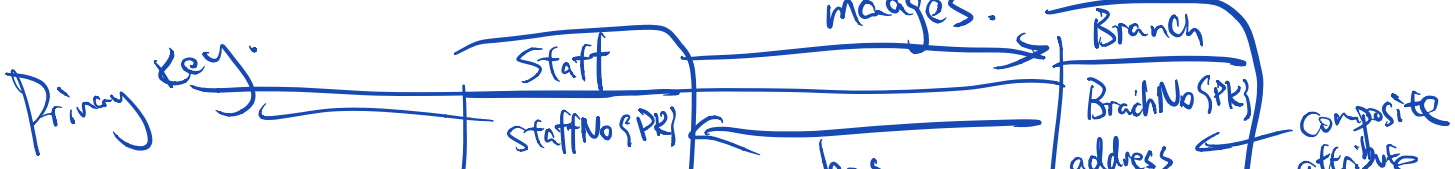
Simple Attribute. / Atomic
Attribute of single component

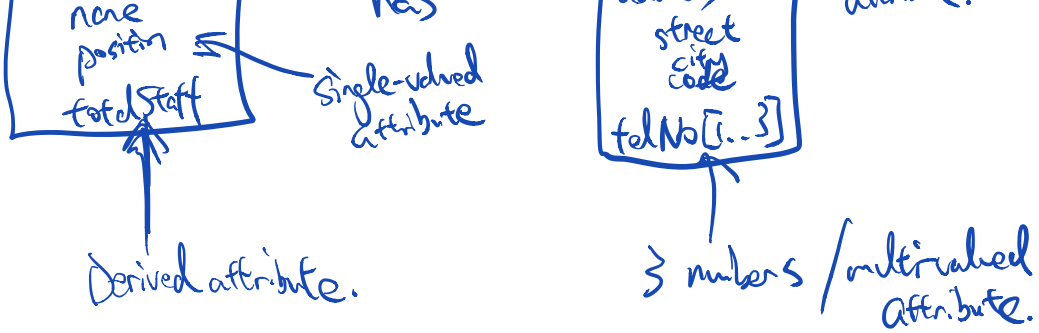
Composite Attribute.
address multiple components

Single-valued Attribute.
holds a single value for each occurence

Multi-valued Attribute. (multiple for each instance)
e.g. telephone numbers for a office.

Derived Attribute.
Duration calculated from lease start to lease end
derivable.

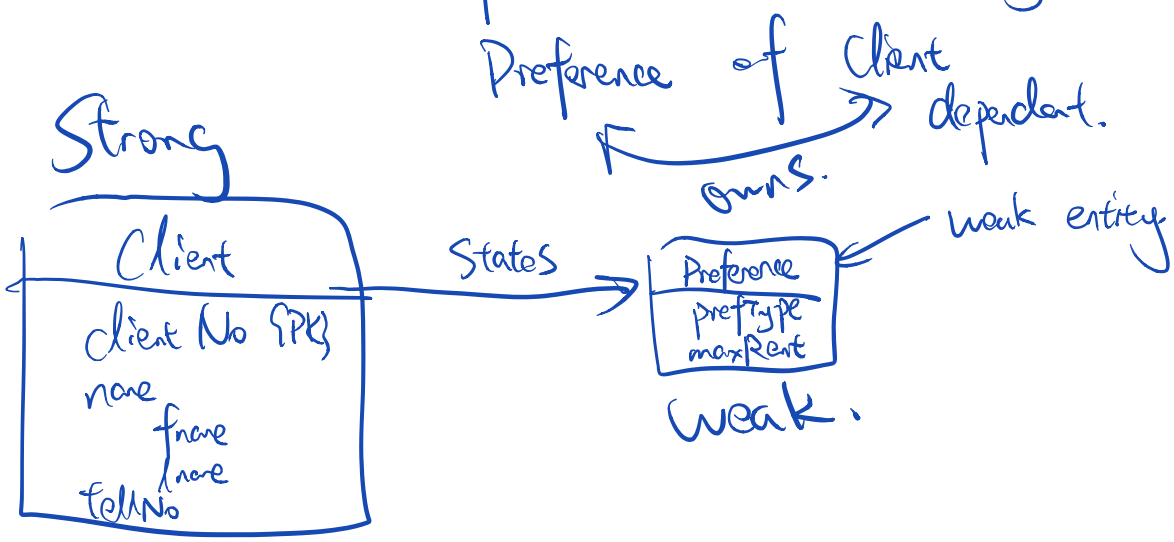




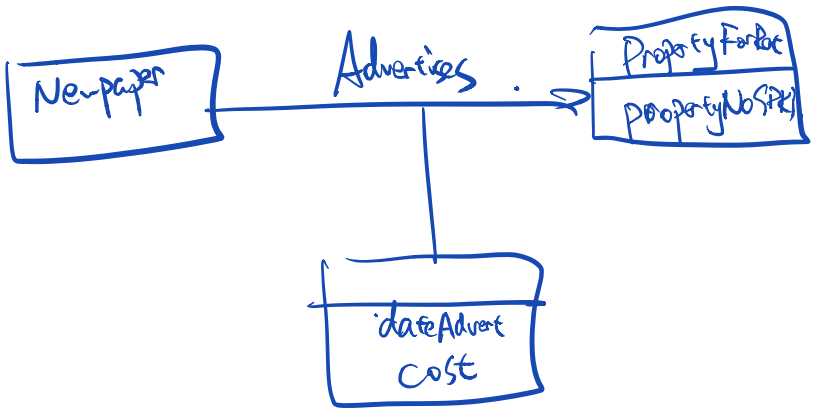
Entity Type

Strong — existence does not depend on other entity

Weak — dependant on other entity



Relationship can have attributes



Structured Constraints.

Property must have an owner.

Branch must have staff.

Main type.: multiplicity constraints.

↓

number or range of possible occurrences of an entity type t.

Represents policies established by company/user

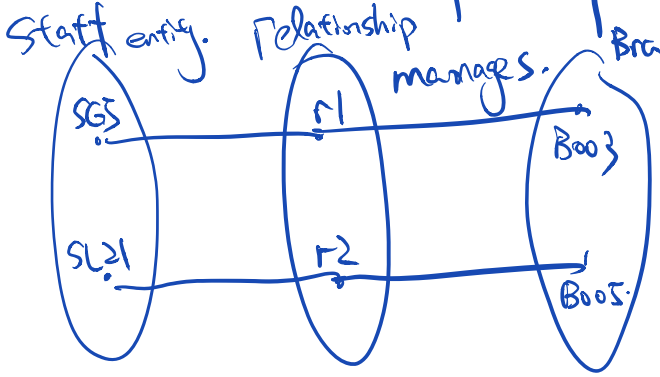
the most common degree of relationships is binary

one-to-one. 1:1

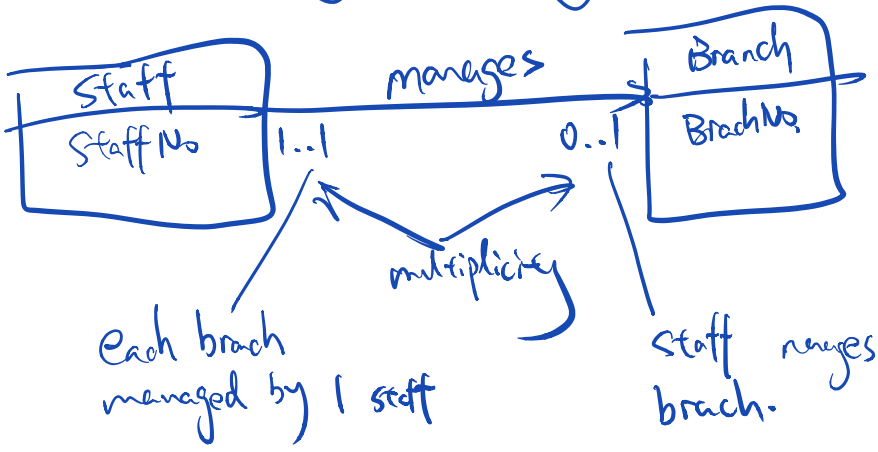
one-to-many 1:*

many-to-many *:*

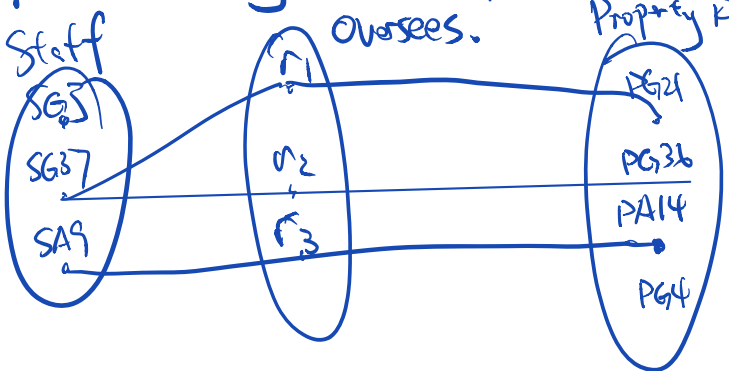
Semantic net example of 1:1 relationship



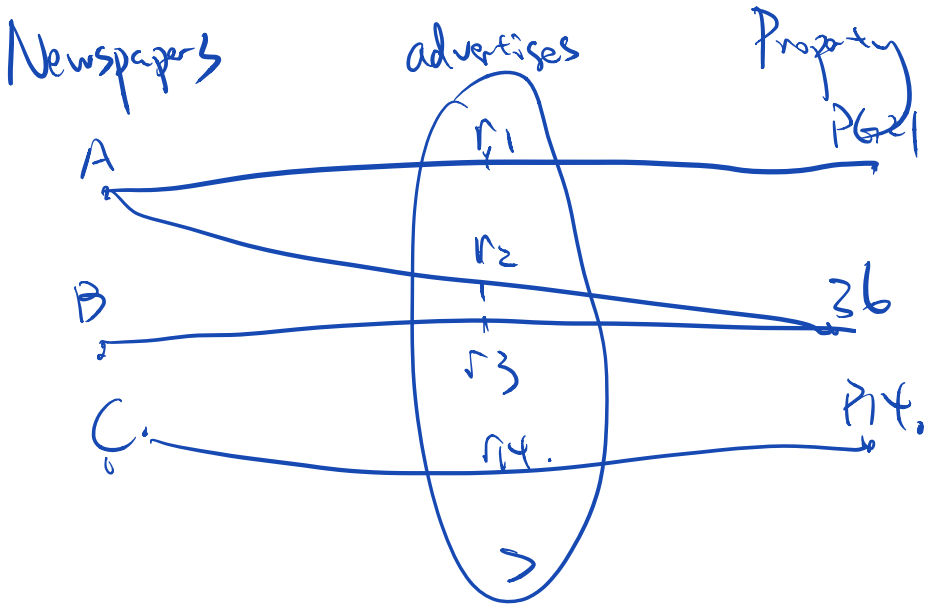
a Staff manages at most one branch.

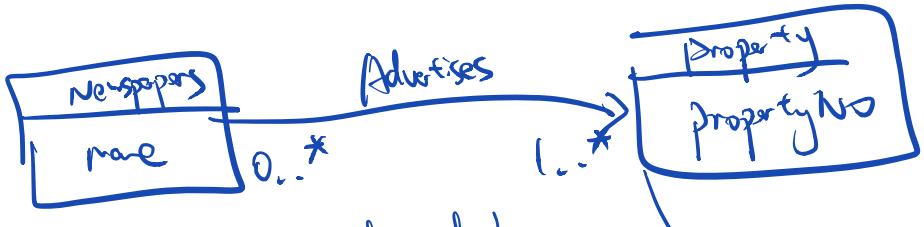


Semantic net of 1-many relationship



Semantic net of many-many relationship

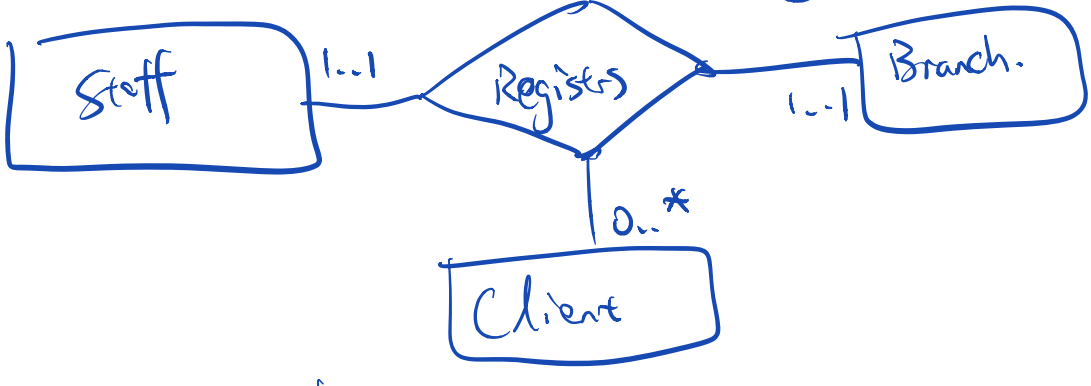




property advertised by 0 or more newspapers.

newspaper advertises 1 or more property

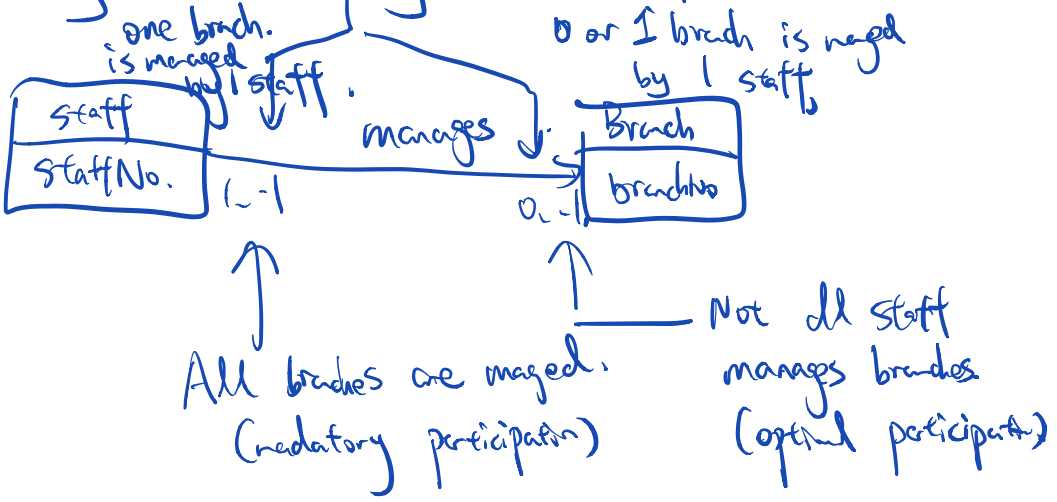
Complex Relationships also have multiplicity



a staff at a branch can register 0 or more clients.

a client at a branch will be reg. by only 1 member of staff

Multiplicity as Cardinality and Participation Constraints



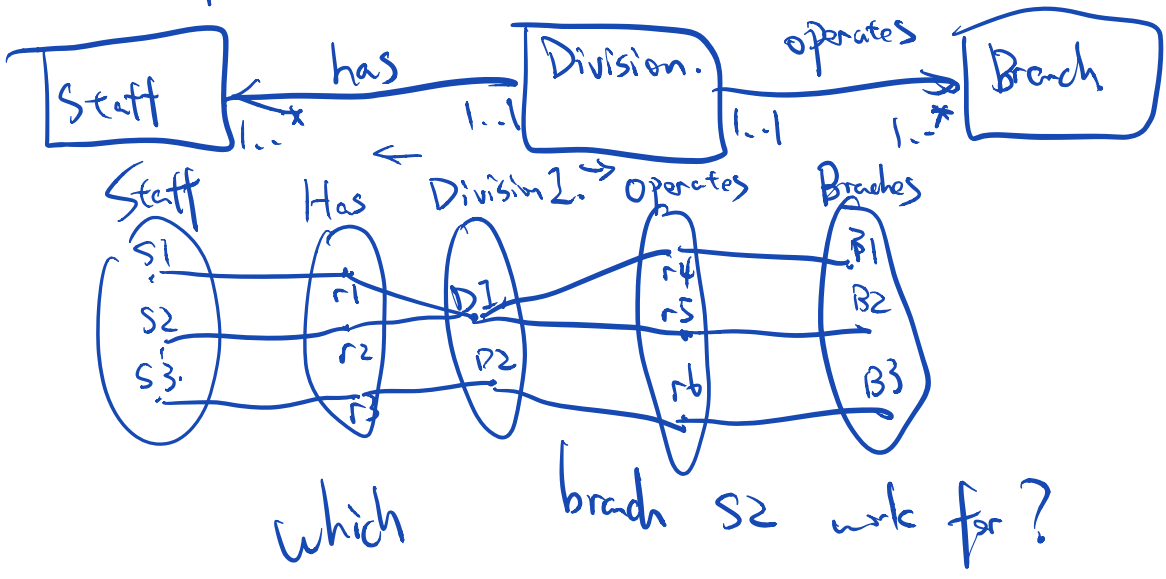
Multiplicity is the restriction

Cardinality - max number of possible relationship instances

Participation - whether some or all instance participate

Connection traps.

Fan trap



which branch S2 work for?

Restructure this:

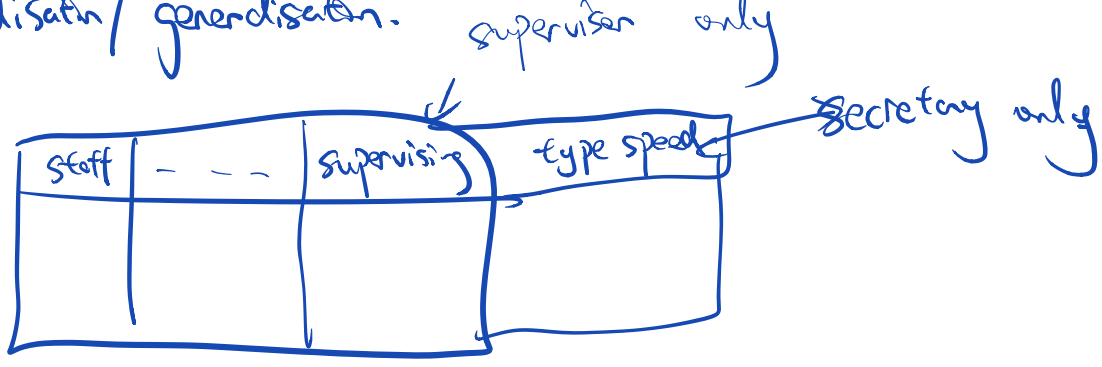
Chasm Trap (Skipped)

Enhanced Entity Relationship Model.

Addition of EER.

* Specialisation / Generalisation
* aggregation
* composition

Specialisation / generalisation.



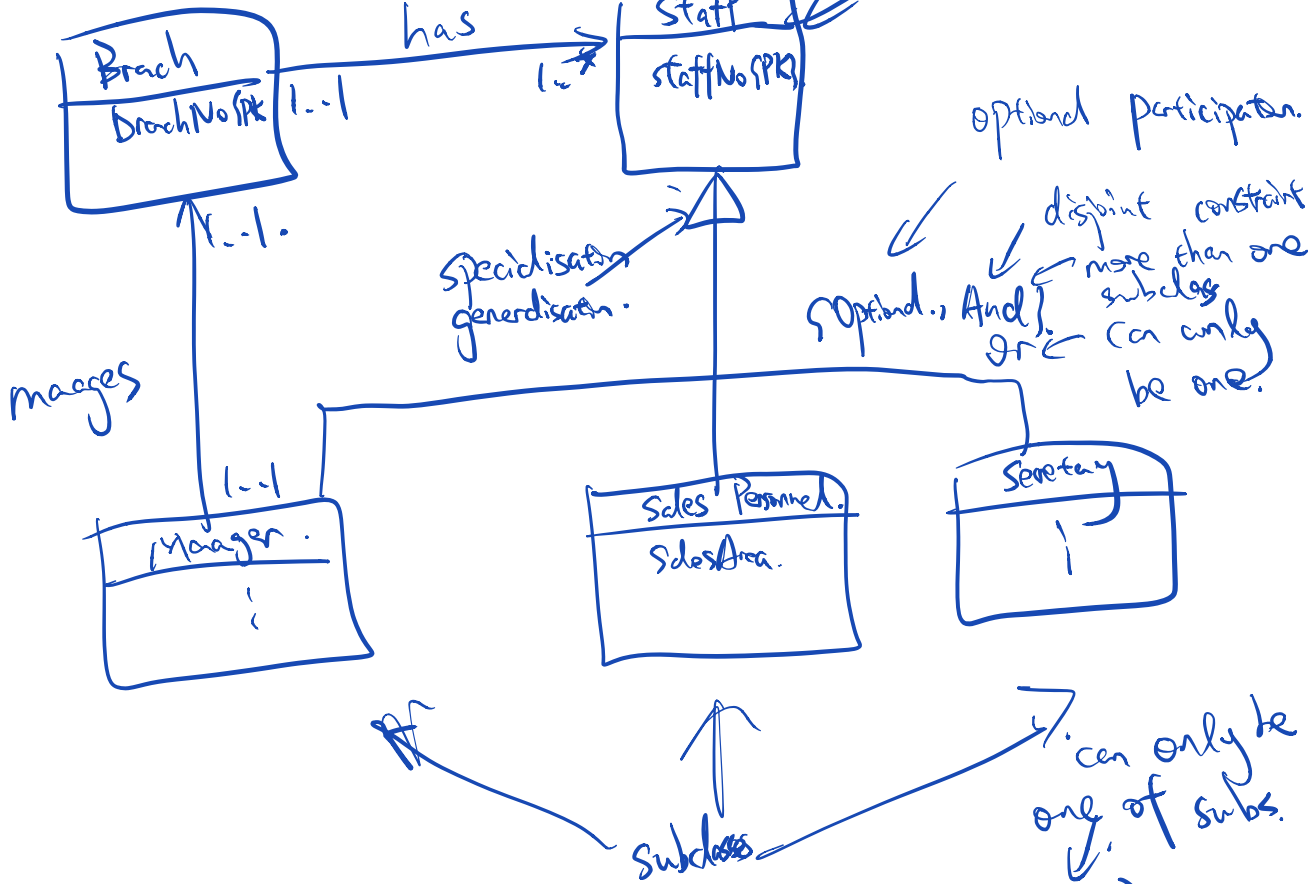
* superclass Staff
* subclass Secretary

superclass / subclass 1:1 relationship.

superclass may contain overlapping / distinct subclasses

not all members of superclass has be a member of subclasses.

— Attribute Inherited



(mandatory / optional , And / Or)

must be one of sub.

can be any of sub or even not sub.

Can be multiple sub at the same time

Logical Design.

* objective of a data model

create an accurate representation of data, relation

* logical design uses the data model to specify data structures.
  - define tables

- translates ER into schema.

Strong entity types.

don't rely on existence of another entity

- create a table for each strong entity
- include simple attributes
- for composite attributes

Client (clientNo, fName--).

Weak entity types

rely on existence of another entity. (e.g. Preference rely on existence of Client)

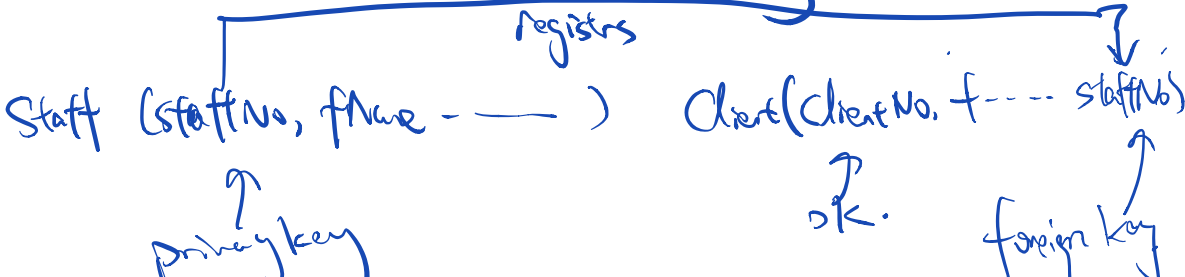
- create a table for each weak entity
- include all simple attributes
- pk partially or fully derived from parent.

Preference (clientNo, prefType, maxRent)

1. Binary Relationships

1- to-many relationship

copy primary key attributes of parent into child as a foreign key



1:1 relationship

3 types of participation.

* mandatory on both
* mandatory on one
* optional on both

Both.

Combine entities into one table.

Choose a pk from attributes that identify one of the original entities uniquely.

Client has one preference and only 1.

Client(clientNo, ——— prefType ——
——staffNo)

staffNo: foreign key

Mandatory on 1 side.

entity with optional is parent.

copy pk for parent and put in table of child.

if the relationship has 1 or more attributes, follow the posting of pk to child

Client(clientNo ——) —states→ Pref(clientNo ——)

clientNo: pk. (Client)

clientNo: pk. (Pref)

foreign key clientNo

refers Client

(c). Optional participation of 1:1 relationship

Identify which is more optional.
and use (b2)

.. relationship. Create table to represent relationship

Complex relationship Types

* create table to represent relationship
* copy of pk of entity related (foreign keys)
* any fk of a 'many' relationship should be part of pk of the new table

Registration (clientNo, branchNo., staff No - -)
pk
fk.

Multi-valued attribute.

create a new table to represent multi-valued

attribute.

Branch (branchNo, ———). Tel (telNo, branchNo)
pk pk. fk.

Superclass, Subclass
↑ parent entity ↑ child entity

Normalisation.

analyses functional dep. between fields
logical design → table.
normalisation → individual table's internal structure

Normal forms (levels of normalisation)
1NF 2NF 3NF BCNF.
higher lvl of integrity →

Normalisation avoids data redundancy

Staff Branch Table.

Cannot insert info until a staff assigned it. (insertion anomaly)

staffNo	fname	branchNo	baddres

delete → delete staff might lose record.

duplicate of branch (modification anomaly)

decomposition removes anomaly.

Normalisation ensure two property of decomposition.

lossless-join property

join decomposed tables would give any instance of original table.

Dependency preservation property.

enforce constraint on the original table by enforcing constraint on the small tables.

Function dependency exaple.

for a specific staffNo, I know the position

position is functionally dependent on staffNo

Not other way round.

staffNo is not —— on position

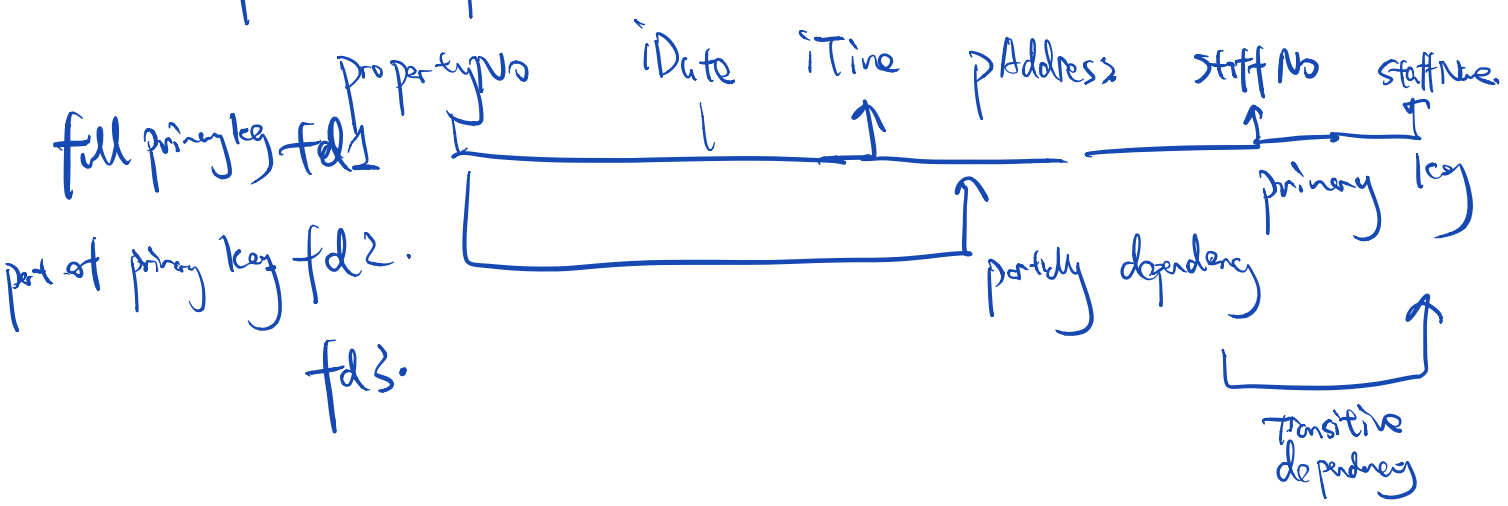
Example

UNF

PG4.	6 La St. Glasgow	18-Oct 22-Apr	SG37 SG14 --
PG16	---	22-Apr 24-Oct	SG14 --

1NF remove repeating groups, identify pk for table.

function dependencies



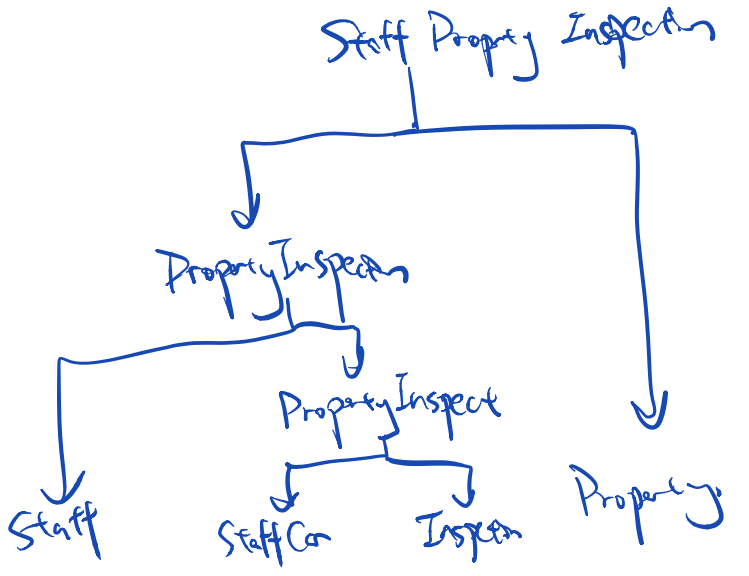
DB at 3rd normal form.

Property (fd2, partially dep.)

Staff (fd3, transitive dep.)

PropertyInspect (fd1, propertyNo, iDate) pk

Still have update anomalies on Car.



1NF no repeating groups

2NF no partial dependences

3NF no transitive dependencies

BCNF every determinant is a candidate key.

UNF — 1NF flatten table.

1NF — 2NF a table with a single attribute as a primary key is a 2NF.

2NF – 3NF transitive dependency.

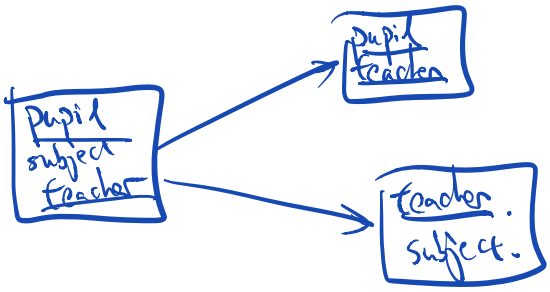
A → B B → C

staffNo → branchNo (via)

branchNo → branchAddres

3NF – BCNF remove any dependencies on any column and set.

it's a BCNF if every determinat is a candidate key



ER. Modeling

Bicycle. Hire Spots. Users.

RefNo
model.
date first service
geo location.
past hire spots / current spots
service history.

guest subscribed users

user number

weak entity Hire

user reg.
bike ref.
hire start.
hire spot.