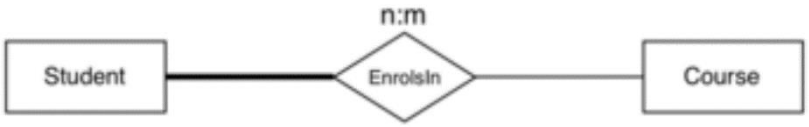
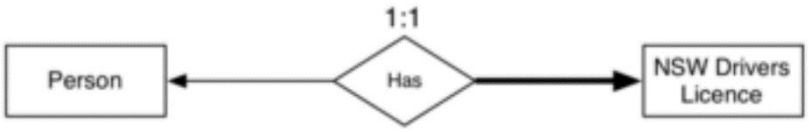
COMP3311 WEEK 02 LECTURE

ER MODEL (cont)



1. Many students can enroll in many courses, all the student must enroll in at least one courses.

2. Many cars registered to car owner, one car owner can have many cars (but at least one car). Each car has only one car owner.



3. One person has one drivers license , each license must be owned by one person.

SUMMARY OF ER

Entities + Relationships + Attributes
Relationship constrains : total / partial, n:m/ 1:n/ 1:1
(inheritance hierarchies, weak entities)

RELATIONAL DATA MODEL

- A collection of inter-connected relations (or tables)
  - Simple, general data modelling formalism
  - Maps easily to file structures (like implementable)
- Mathematical (relation, tuple, attribute) + data-oriented (table, record, field/column)
- In relational model, ROWs represent records + COLUMNs represent attributes.

1. RELATION = Name (unique within a given relation) + A set of attributes (column headings)
   a. Each relation has a KEY (subset of attributes unique for each tuple)

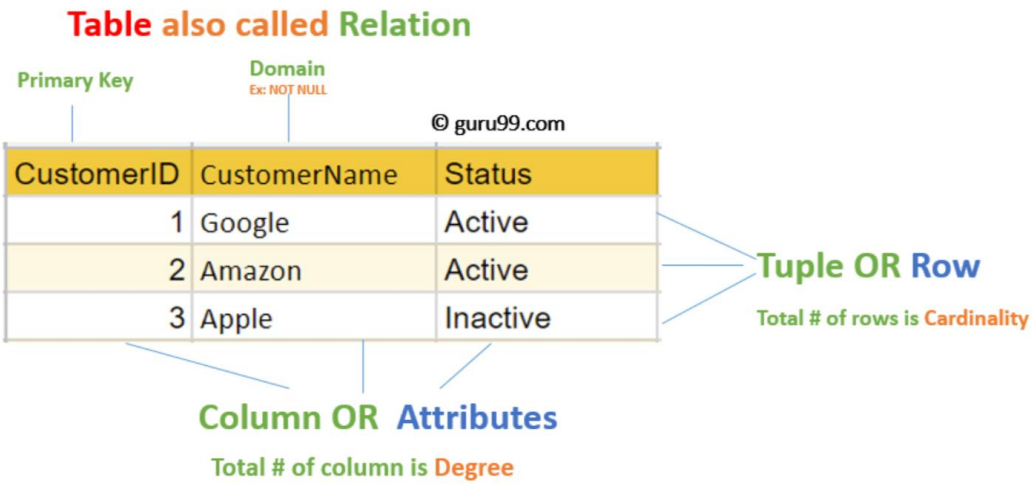
   A relation is nothing but a table of values.

   Every row in the table represents a collection of related date values (these rows in the table denote a real-world entity or relationship)

   The data are represented as a set of relations.

2. ATTRIBUTES = Name + An associated domain
   a. Attribute values are ATOMIC (no composite or multi-valued attributes)
   b. a distinguished value NULL belongs to all domains

   Attribute is each column in a table. They are the properties which define a relation.



Database schema = a collection of RELATION SCHEMAS

Database (instance) = a collection of relation instances

DATABASE MANAGEMENT SYSTEMS

(Database management system is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data)

1. Relational model is a mathematical construct
   i. giving a representation for data structures
   ii. with constrains as logic formulae on relations/tuples
   iii. an algebra for manipulating (handle or control) relations/tuples
2. Relational Database Management systems (RDBMSs)
   i. RDBMSs is a collection of programs and capabilities that enable IT teams and others to create, update, administer and otherwise interact with a "relational model". Mostly use Structured query language (SQL) to access the database.
   ii. Provide an implementation of the relational model
   iii. Using SQL as language for: data definition, query, update
3. Approximations made by SQL
   i. Relations are not required to have a key
   ii. Relations are bags rather than sets

DBMS TERMINOLOGY

Many DBMSs have multiple namespaces:

- DBMS-level – database names must be unique
- Database-level – schema names must be unique
- Schema-level – table names must be unique
- Table-level – attribute name must be unique

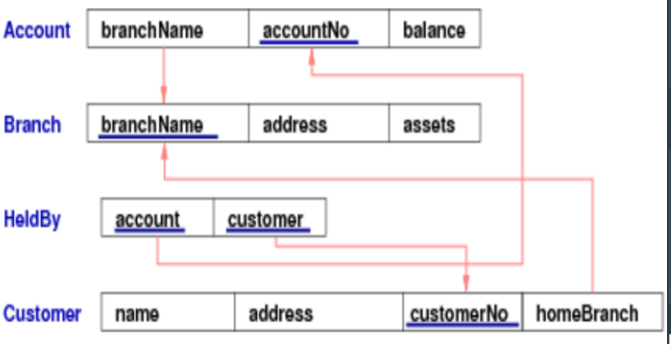
Sometimes it is convenient to use same name in several tables.

->so we distinguish which attribute we mean using qualified names

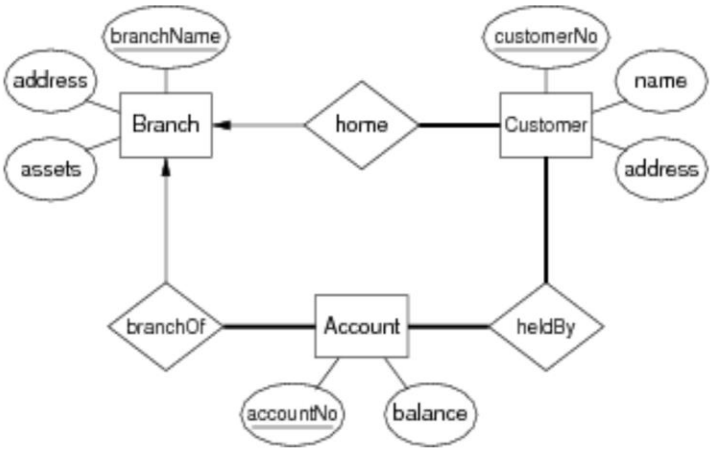
(eg. Account.branchname && Branch.branchname)

Example Database Schema

Schema with 4 relations:




```
--in MANY-TO-ONE relationship
--create table for many side contains the other side
create table Account (
    AccountNo   integer,
    Balance     float not null,
    BranchOf varchar(80) not null references to Branch(BranchName),
    primary key (AccountNo)
);
create table Customer (
    CustomerNo  integer,
    name        varchar(50) not null,
    address     text not null,
    home varchar(80) not null references to Branch(BranchName),
    primary key (CustomerNo)
);
```



Example Database (Instance)

Account

branchName	accountNo	balance
Downtown	A–101	500
Mianus	A–215	700
Perryridge	A–102	400
Round Hill	A–305	350
Brighton	A–201	900
Redwood	A–222	700

Branch

branchName	address	assets
Downtown	Brooklyn	9000000
Redwood	Palo Alto	2100000
Perryridge	Horseneck	1700000
Mianus	Horseneck	400000
Round Hill	Horseneck	8000000
North Town	Rye	3700000
Brighton	Brooklyn	7100000

Customer

name	address	customerNo	homeBranch
Smith	Rye	1234567	Mianus
Jones	Palo Alto	9876543	Redwood
Smith	Brooklyn	1313131	Downtown
Curry	Rye	1111111	Mianus

Depositor

account	customer
A–101	1313131
A–215	1111111
A–102	1313131
A–305	1234567
A–201	9876543
A–222	1111111
A–102	1234567

INTEGRITY CONSTRAINSTS

- Relations are used to represents ENTITIES and RELATIONSHIPS
- Domain limits the set of values that attributes can take
- To represents real-world problems, need to describe:
  - What values are/ are not allowed
  - What combinations of values are/ are not allowed
- Constraints are logical statements that do this:
  - Domain, key, entity integrity, referential integrity…

---

1. *NOT NULL – ensures that a column cannot have a NULL value*
2. *UNIQUE – ensures that all values in a column are different*
3. *PRIMARY KEY – not null + unique. Uniquely identifies each row in a table*
4. *FOREIGN KEY – uniquely identifies a row/record in another table*
5. *CHECK – ensures that all values in a column satisfies a specific condition*
6. *DEFAULT – sets a default value for a column when no value is specified*
7. *INDEX – used to create and retrieve date from the database very quickly*

---

1) Domain constraints example:
    a. “Employee.age” attribute is typically defined as integer
    b. Better modeled by adding extra constraint (15<age<66)

```
create table Employee (
    age     integer,
    age     integer check (between 15 to 66),
);
```

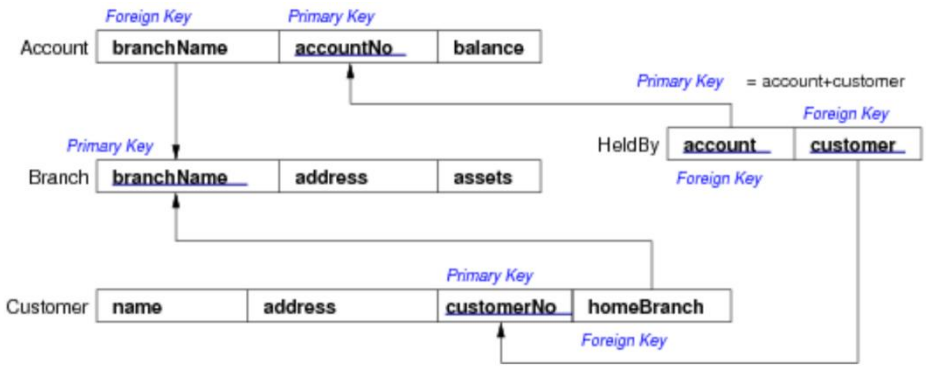
    c. NULL satisfies all domain constraints (except (not null))
2) Key constraints example:
    a. Student (id, …) is guaranteed unique

```
create table students (
    ID  integer not null unique,
    lastName varchar(255) not null,
    firstName varchar(255),
    primary key(ID) --primary key is unique and not null
);
```

    b. Class (…, day, location, time, …) is unique
3) Entity integrity example:
    a. Class (…, mon, 2pm, lyre, …) is well defined
    b. Class (…, NULL, 2pm, lyre, …) is not well defined

REFERENTIAL INTEGRITY

- Describe references between relations (tables)
- Are related to notion of a foreign key.
    - The foreign key may be null and may have the same value but:
        - The foreign key value must match a record in the table it is referring to
    - Tables that do not follow this are violating the referential integrity rule



A set of attributes F in relation R1 is a foreign key for R2 if:
- the attributes in F correspond to the primary key of R2
- the value of F in each tuple of R1
Either occurs as a primary key in in R2
Or is entirely NULL

RELATIONAL DATABASES

A relational database schema is

- A set of relation schemas {R1, R2, …, Rn}, and
- A set of integrity constraints

A relational database instance is

- A set of relation instances {r1(R1), r2(R2), …}
- Where all of the integrity constraints are satisfied

One of the MOST important functions of relational DBMS: **ensure that all data in the database satisfies constraints**

DESCRIBING RELATIONAL SCHEMAS

We need a formalism to express relational schemas

SQL provides a Data Definition Language (DDL) for this.

```
CREATE TABLE TableName (
        attrName1       domain1         constraints1,
        attrName2       domain2         constraints2,
        …
        PRIMARY KEY (attr1, attr2, …)
        FOREIGN KET (attr5, attr6, …) REFERENCES OtherTable(attr8, attr9)
);
```