1 INFS

INFS1200

Introduction to Information Systems

Library Site: http://www.library.uq.edu.au/lr/INFS1200

Week 1

Primary Goals

1. Extract information systems requirements to create basic conceptual models
2. Map basic conceptual data models to relational database schema
3. Reason with the logical foundation of the relational data model and the fundamental principles of correct relational database design
4. Express natural language queries using the SQL language
5. Explain key security concepts related to database control measures and SQL injection
6. Construct a small-scale information system in a relational database management system.

Data: facts and statistics collected together for reference or analysis, typically digitised.

Information and Systems:
Information: Data put into meaningful and useful context and communicated to recipients who use it to make decisions

System: A set of components that interact to accomplish some purpose.

Data management

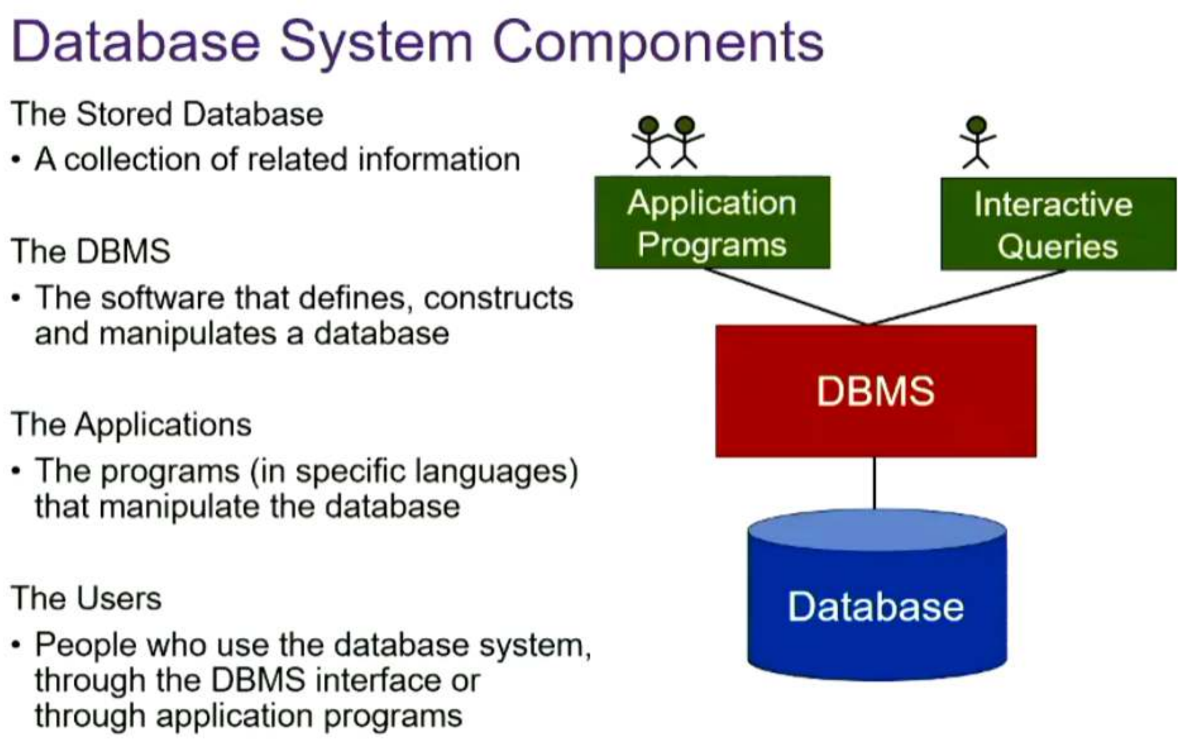
- Essential skill for future workplace
- **Banking** is an example of an application area where data plays a central role

Database

- Collection of related data or known facts that:
    - Represents some aspect of the real world (mini-world) or the universe of discourse(UoD)
    - Is a logically coherent collection of data with some inherent meaning
    - Is designed, built, and populated with data for a specific purpose for an intended group of users and some preconceived applications.

DBMS - Database Management System

- Software system that facilities the processes of defining, constructing, manipulating and sharing databases among various users and applications.
    - Defining a database includes specifying the types, structures and constraints for the data.
    - Constructing the database is the process of storing the data on some storage medium that is controlled by the DBMS.
    - Manipulating a database includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data.
    - Sharing a database allows multiple users and programs to access the database simultaneously.



Database System Components

The Stored Database
- A collection of related information

The DBMS
- The software that defines, constructs and manipulates a database

The Applications
- The programs (in specific languages) that manipulate the database

The Users
- People who use the database system, through the DBMS interface or through application programs

Application Programs
Interactive Queries
DBMS
Database

Database Management System provides a facility to recover from hardware and software failures through its backup and recovery sub-system.

Data Independence via the Three-Schema Architecture
Data Independence: Ability to change the schema at one level of a database system without having to change the schema at the next higher level.

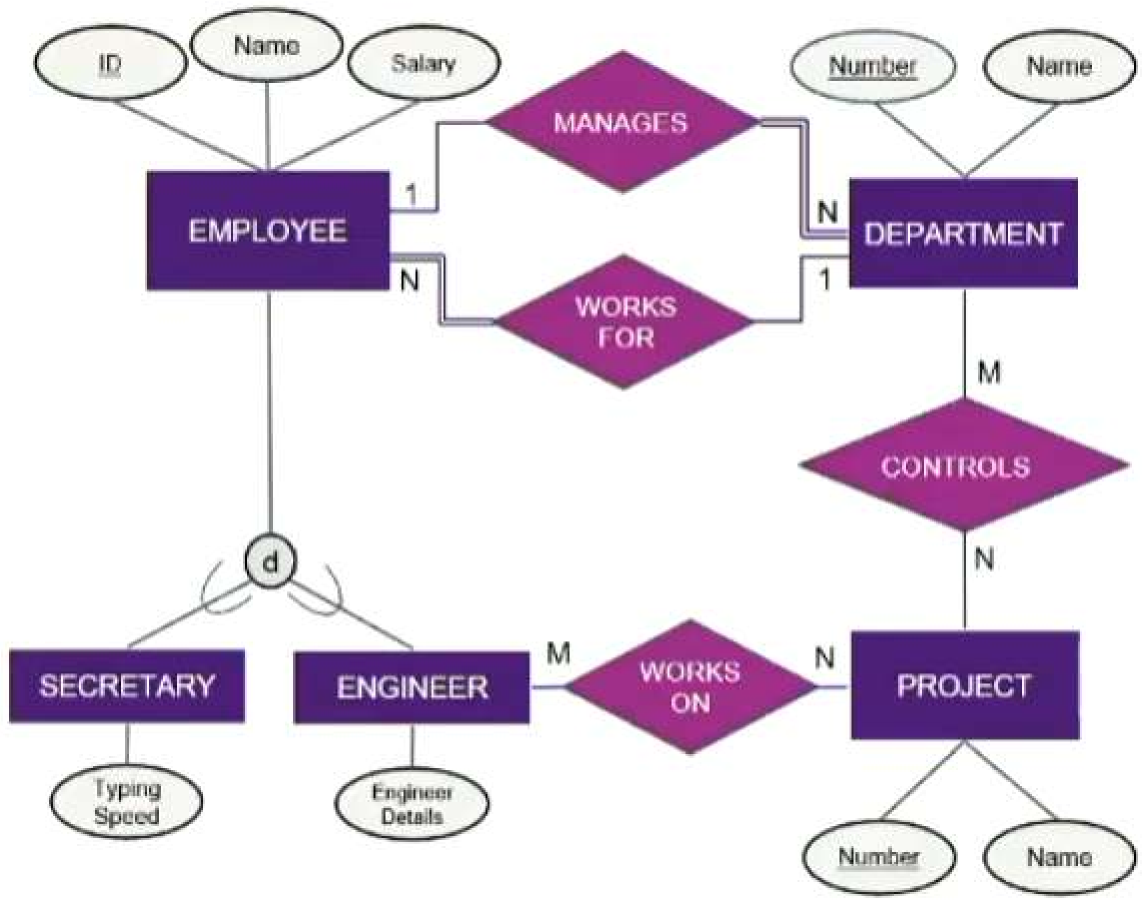
1. Logical Data Independence: Ability to change the conceptual schema without changing external views or applications.
2. Physical Data Independence: Ability to modify physical schema w/o changing logical schema.

Week 2

Conceptual Database Design
A model is never perfect.

The Entity-Relationship (ER) model at its core provides a graphical representation of data entities.

ER models assist database designers to define a project's scope and requirements for clients and businesses



(ER model example of a company)

Entities
- Entities, Entity Sets and Entity Types
- Attributes, Keys and value sets
- Weak entities

Relationships
- Relationship types and sets

- Relationship Degree
- Roles and Recursive Relationships
- Relationship constraints
- Attributes of Relationship types

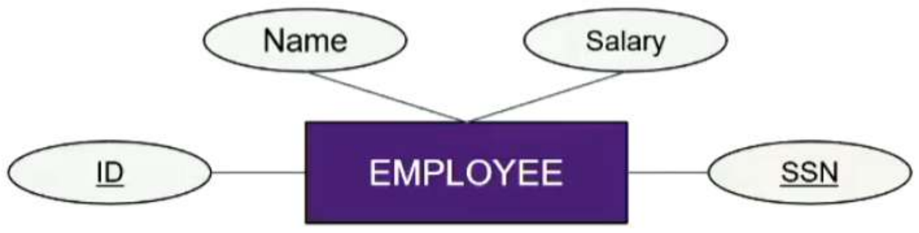
Entity
A physical or conceptual object which has data associated with it

May have various attributes associated with it

Same entity can have different attributes depending on the specific requirements of that system.

An entity type provides a format for the data which needs to be recorded to represent a particular entity. The entity type is described by its name and attributes.

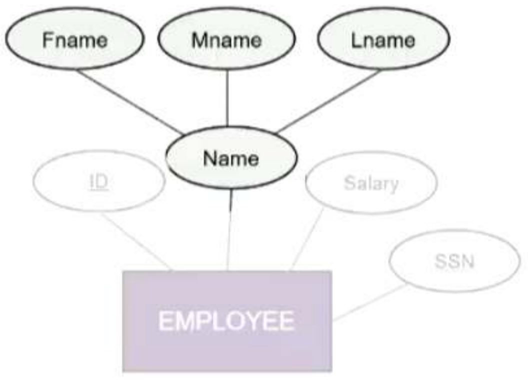
In the ER model:
- An entity type is represented as a rectangular box.
- Attribute names are represented by ovals attached to their entity type.



All entity types have at least one key constraint.

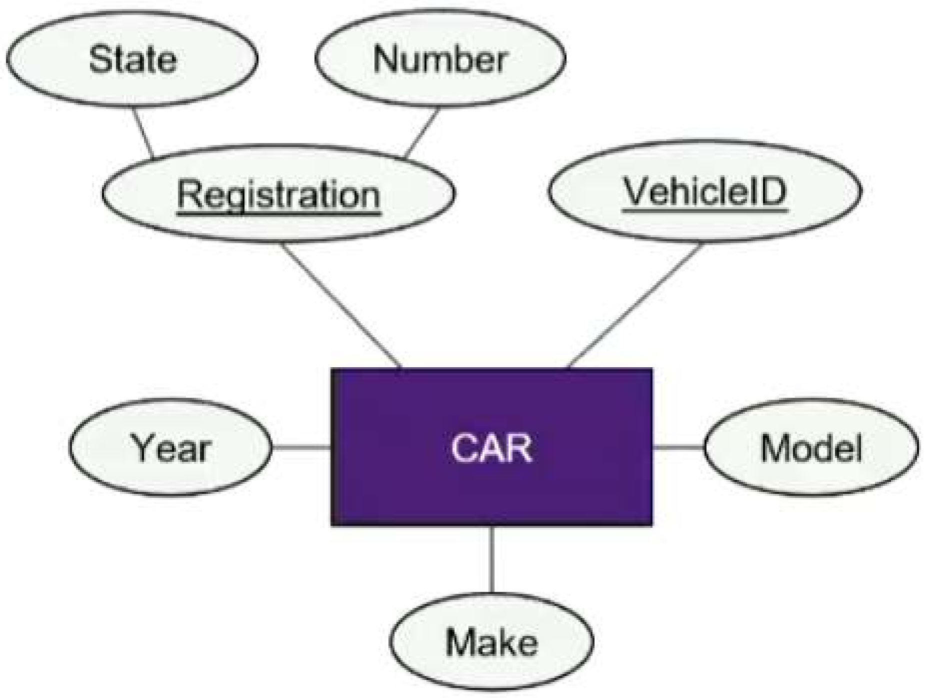
In the ER model:
- Key attribute is underlined inside the oval
- Key must hold for every possible extension of the tnt type
- Multiple keys are possible.

Composite attributes and be divided into smaller parts which represent simple attributes with independent meaning.

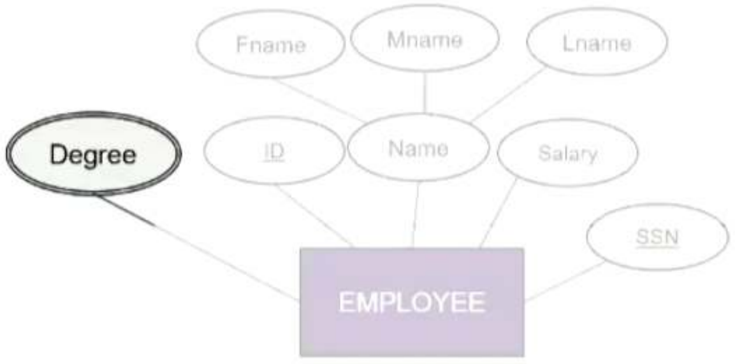


Several Attribute Keys

- A composite key attribute is also a unique identifier for each entity instance.
- A composite key is a combination of simple attributes which make up the composite key that must be unique.

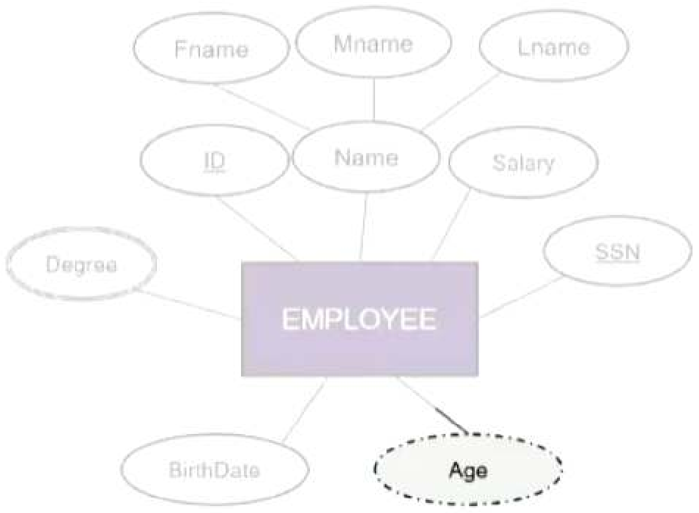


Multivalued attributes are shown with double-lined ovals and can have multiple values.



Derived Attributes
Are shown with an oval with a dotted line.

Derived from related attributes (eg. age from birthdate)



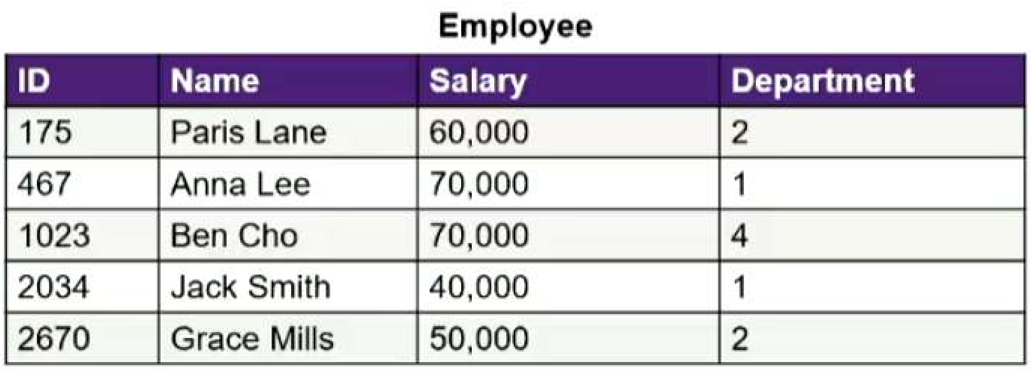
Value Sets of Attributes

- employeeAge: integers between 21 & 65
- vehicleRegistrationNum: String of 3 alphabets followed by 3 integers Value sets are not displayed on the ER diagram

A particular entity may not have an applicable value for an attribute.
A null value may be used for representing this.

- tertiaryDegree: Not applicable for a person with no university education
- homePhone: not known if it exists
- height: missing

Entity Set
The collection of all entities of a particular entity type in the database at any point in time is referred to as an entity set.
An entity can be mapped to a table.

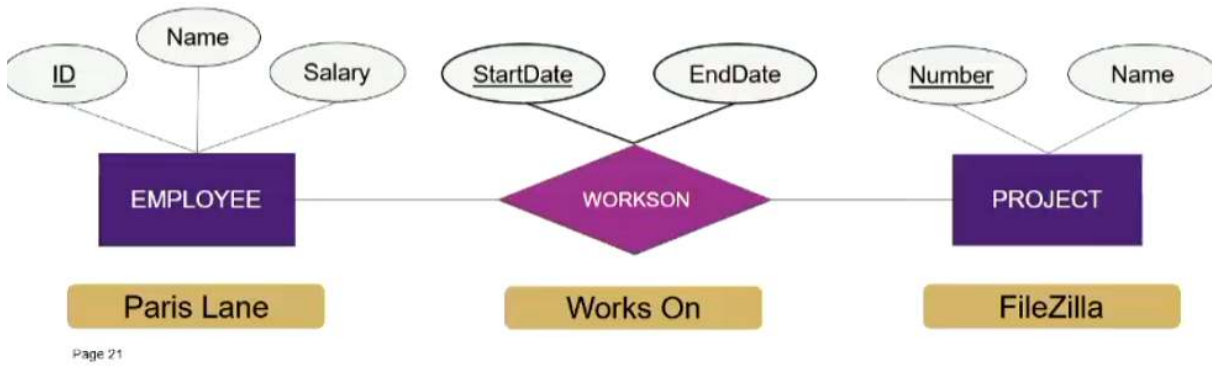


Employee

ID	Name	Salary	Department
175	Paris Lane	60,000	2
467	Anna Lee	70,000	1
1023	Ben Cho	70,000	4
2034	Jack Smith	40,000	1
2670	Grace Mills	50,000	2

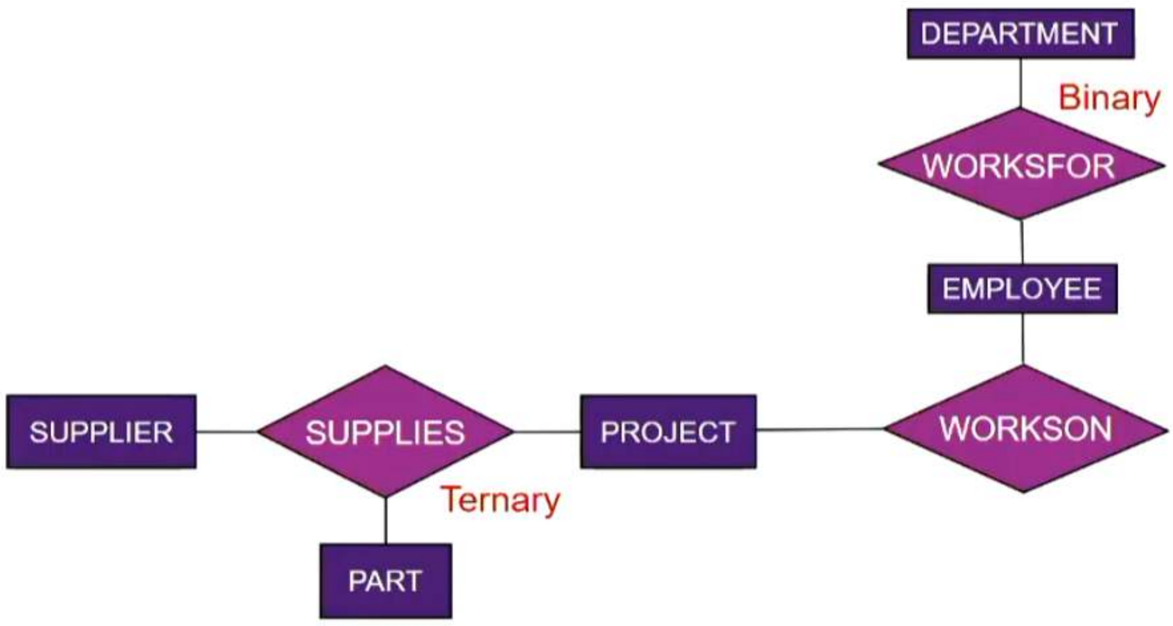
Relationship Types
A relationship is an association among two or more entities
A relationship type defines the relationship

- In the ER model, relationships are represented using a diamond that is connected to the associated entity types.
- A relationship type may have descriptive attributes.
- A relationship type may have key attributes.

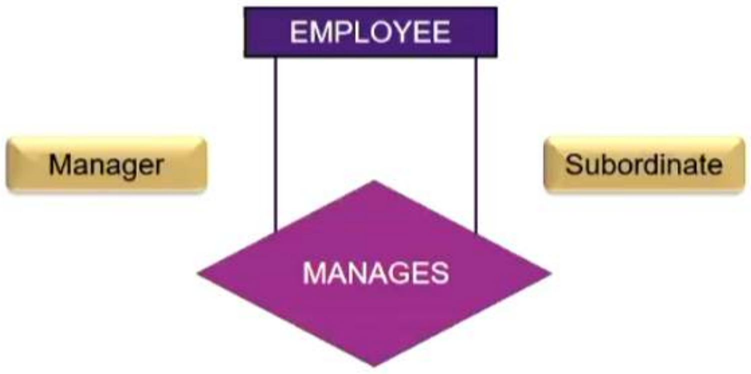


Relationship Degree
The degree of a relationship type is the number of participating entity types

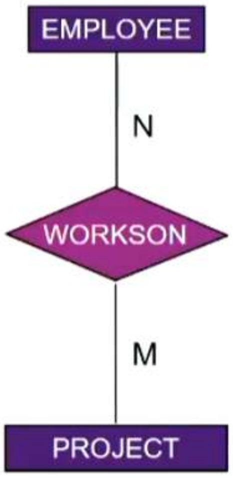
- 2 entities: Binary Relationship
- 3 entities: Ternary Relationship
- n entities: N-ary Relationship (3+ Entities)

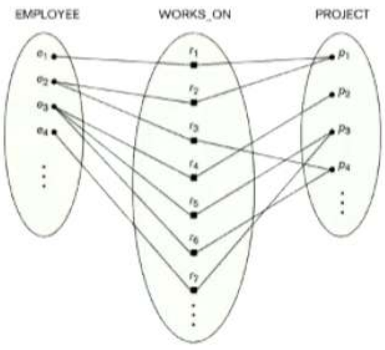


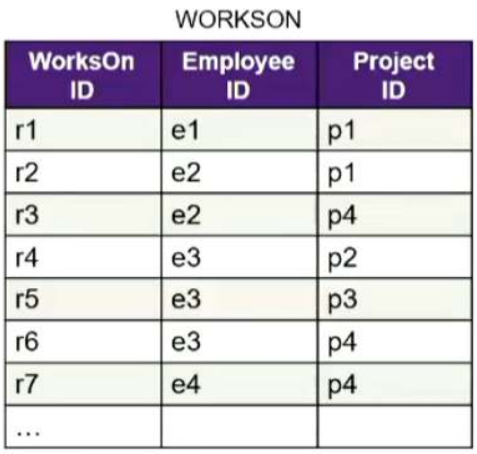
Recursive Relationships
Some entity types can participate more than once in the same relationship type under different “roles”.



Relationship Set
The collection of all relationships of a particular relationship type in the database at any point in time is referred to as a relationship set.
There are different methods of representing relationship sets.



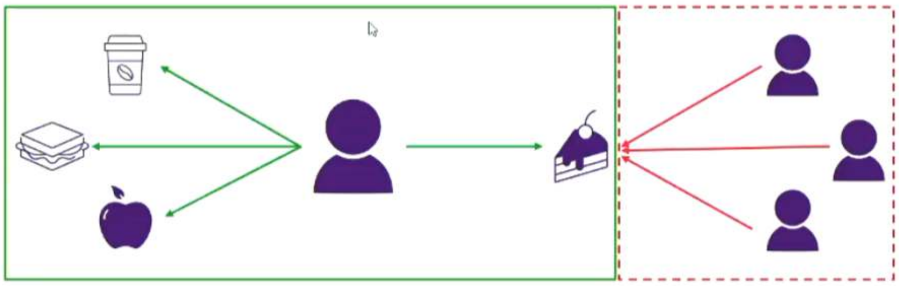



WORKSON

WorksOn ID	Employee ID	Project ID
r1	e1	p1
r2	e2	p1
r3	e2	p4
r4	e3	p2
r5	e3	p3
r6	e3	p4
r7	e4	p4
…		

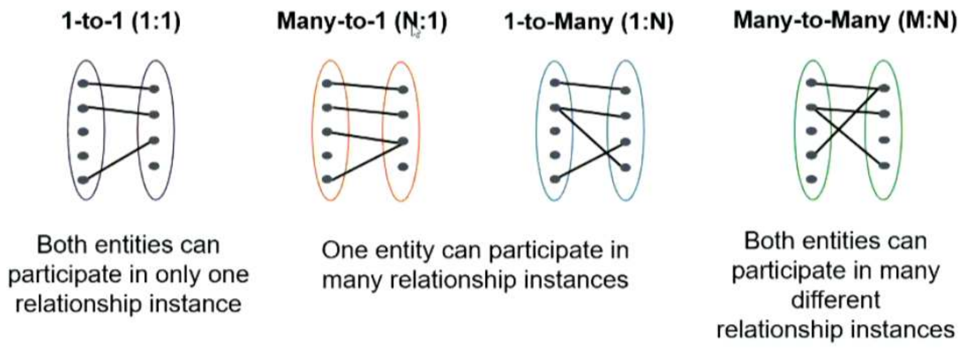
Relationship Constraints
Constraints on the relationship type limit the possible combination of entities that may participate in the corresponding relationship set.

A cardinality ratio for a relationship set specifies the number of relationships in the set that an entity can participate in.


One particular customer can purchase **many** different items
…however…
n individual item (E.g. a cake) cannot be purchased by several different customers.

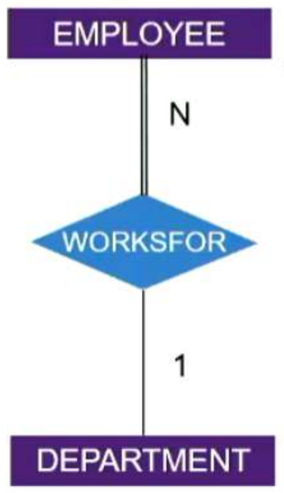
Cardinality Ratio

A **cardinality ratio** for a relationship set specifies the number of relationships in the set that an entity can participate in.


1-to-1 (1:1)
Many-to-1 (N:1)
1-to-Many (1:N)
Many-to-Many (M:N)
Both entities can participate in only one relationship instance
One entity can participate in many relationship instances
Both entities can participate in many different relationship instances

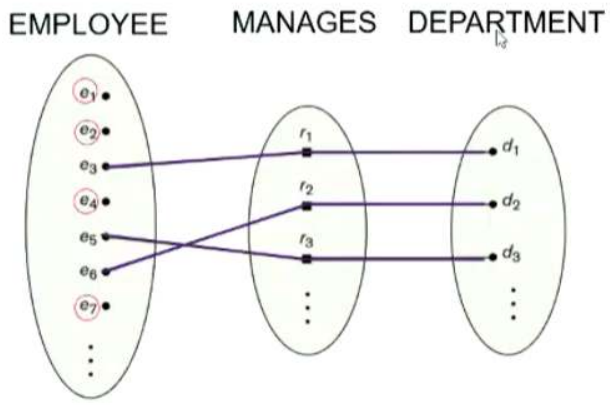
Existence Dependency

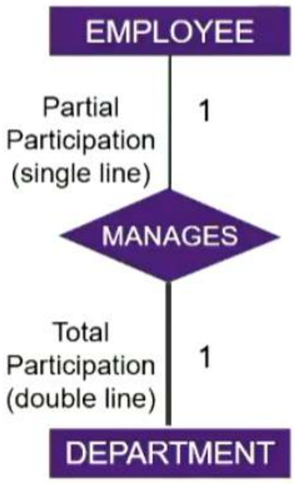
Indicates whether the existence of an entity depends on its relationship to another entity.



Every employee must work for a department.

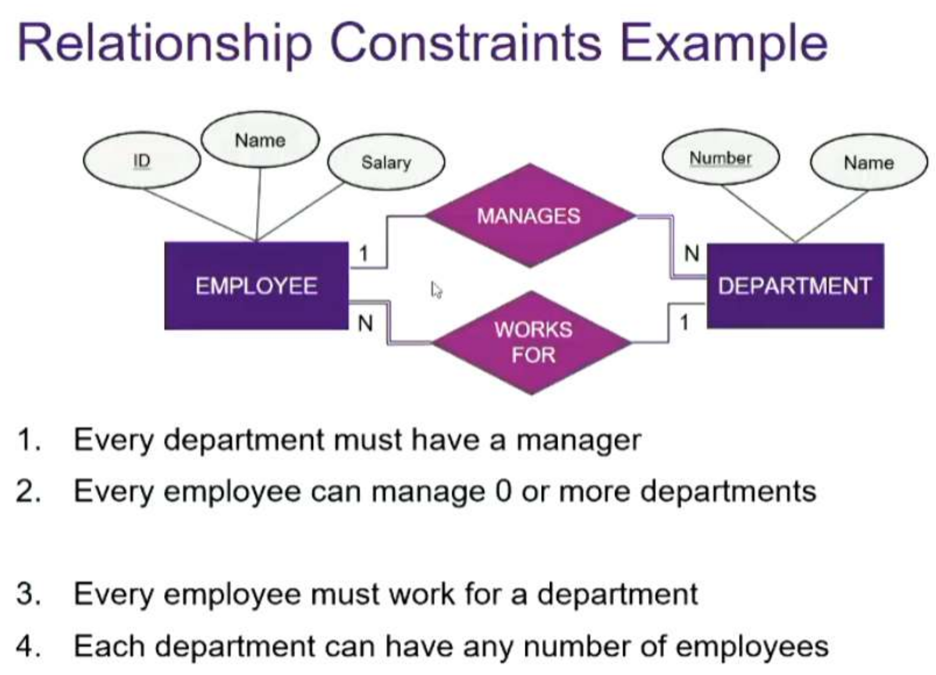
Employee is existentially dependent on **Department** via the **WORKSFOR** relationship type.




EMPLOYEE
Partial Participation (single line)
1
MANAGES
Total Participation (double line)
1
DEPARTMENT

Every department must have a manger.

Department is existentially dependent on **Employees** via the **Manages** relationship type.

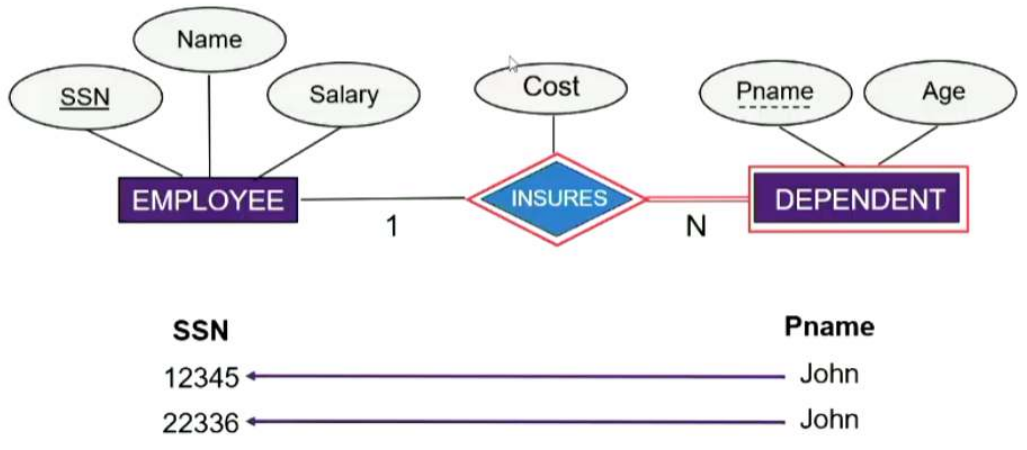


Relationship Constraints Example

1. Every department must have a manager
2. Every employee can manage 0 or more departments

3. Every employee must work for a department
4. Each department can have any number of employees

Weak Entities
Entity types that do not have key attributes of their own are called weak entities.

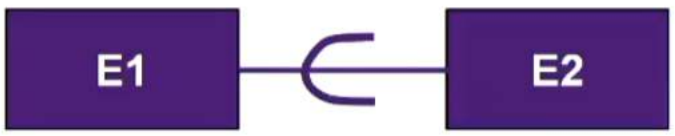


SSN	Pname
12345	John
22336	John

Can be identified uniquely only be considering the primary key of another owner entity and its own partial key, which is underlined with a dotted line
The relationship type that relates a weak entity to its owner is called the identifying relationship. A weak entity and its identifying relationship are represented with double lines.

The Enhanced ER (EER) Model
Entity Type is called class in EER
Entities in the same class have the same attributes
Can be a superclass or a subclass

Every entity in a subclass is a member of its superclass(es)

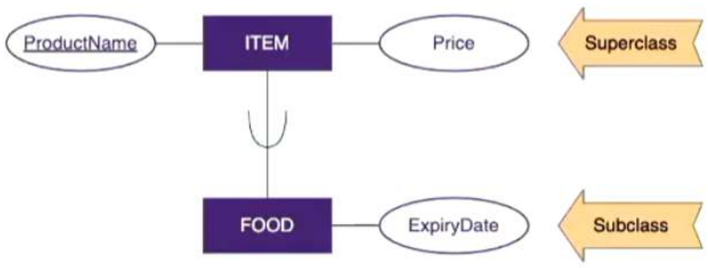


E1 is the subclass of E2, as indicated by the direction of the ⊂.

Motivating Example

You may purchase a variety of different items from a supermarket but what do all these items have in common? It's more than likely a **product name** and a **price**.

For some type of items though, say for example, food items there may be extra associated data such as an **expiry date**.



In this scenario, **"item"** is what we could call a **super class**. It captures the data common for a variety of different objects. A **"food"** item, which is just an extension of a regular item, is then what we would call a **subclass**.

Specialisation/Generalisation
There are two ways super/subclasses can be identified
Specialisation

- Define a number of subclasses of an entity type
- Each subclass contains a subset of entities from the superclass
- A subclass is defined based on more specific distinguishing characteristic on entities of the superclass

Generalisation

- Abstraction is the process of ignoring differences amongst some subclasses and generalising them into a superclass.

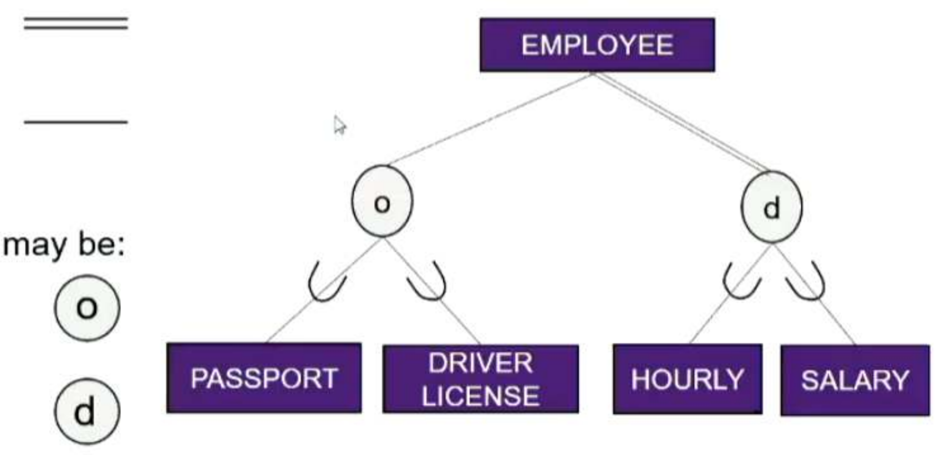
Constraints

Specialisation may be:

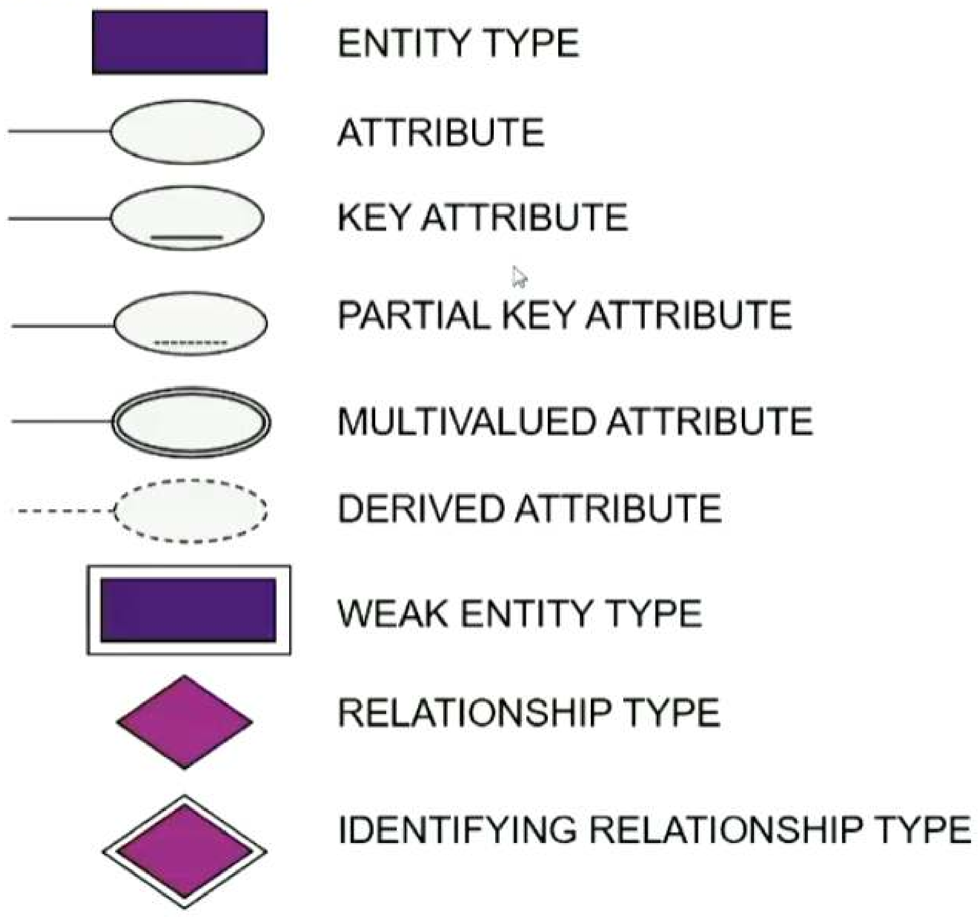
- total
- partial

Subclass sets may be:

- overlapping
- disjoint



Notation Guide

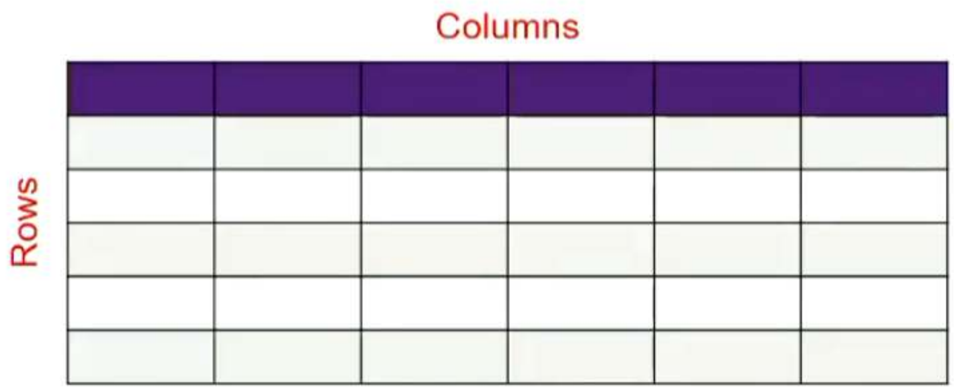


Week 3

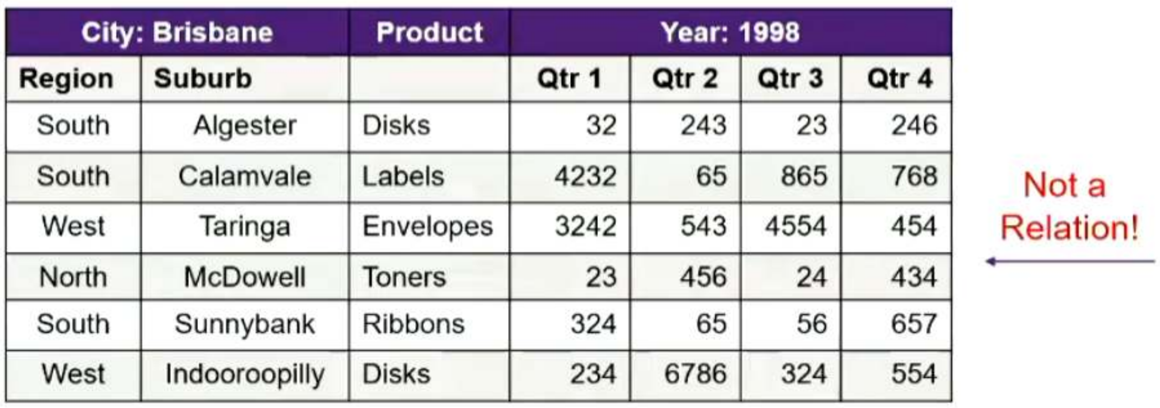
Relational Model
- Introduced by E.F Codd in 1970
- Many DBMS rely on this model

Four basic concepts:
- Relations
- Attributes
- Domains
- Tuples

A relation is a set of records, similar to a table with columns and rows



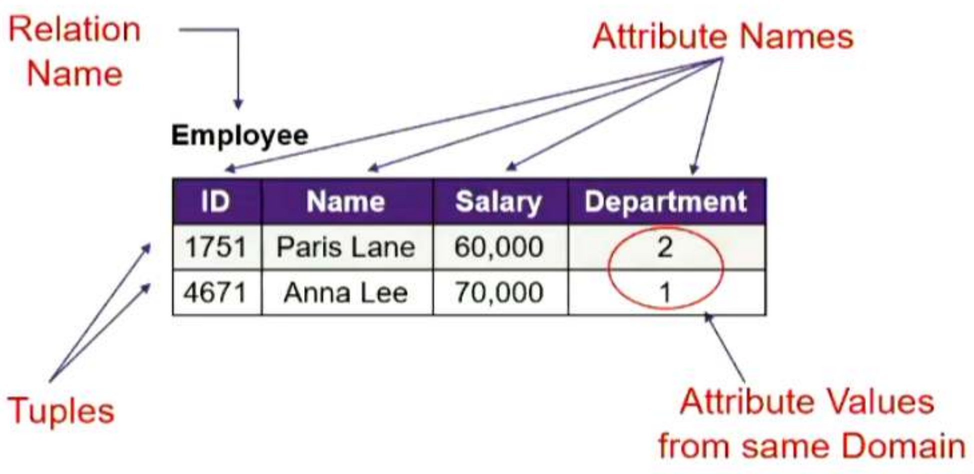
Every relation is as table
Not every table is a relation
Relations have specific properties, based on the mathematical set theory.



City: Brisbane		Product	Year: 1998			
Region	**Suburb**		**Qtr 1**	**Qtr 2**	**Qtr 3**	**Qtr 4**
South	Algester	Disks	32	243	23	246
South	Calamvale	Labels	4232	65	865	768
West	Taringa	Envelopes	3242	543	4554	454
North	McDowell	Toners	23	456	24	434
South	Sunnybank	Ribbons	324	65	56	657
West	Indooroopilly	Disks	234	6786	324	554

Not a Relation!

Relation Components



Employee

ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1

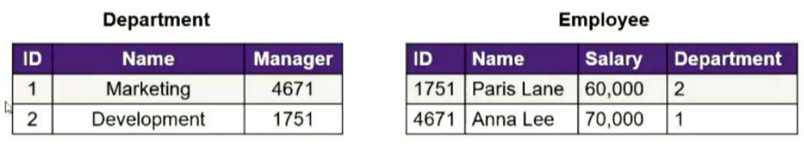
Domain Types
A domain D is a set of atomic values
An atomic value is indivisible (as far as the relational data model is concerned)

Each domain has a data type or format

- Integers
- Numbers and currency
- FIxed or variable length strings
- Date, timestamp
- Sub-range from a data type
- Enumerated data type
- Australian phone numbers
- Car rego numbers

Attributes
Each attribute A is the name of a role played by some domain D in the relation named R
The number of attributes in a relation R is called the degree of R

Domain/Attribute Restrictions
Same attribute name does not necessarily mean same domain.



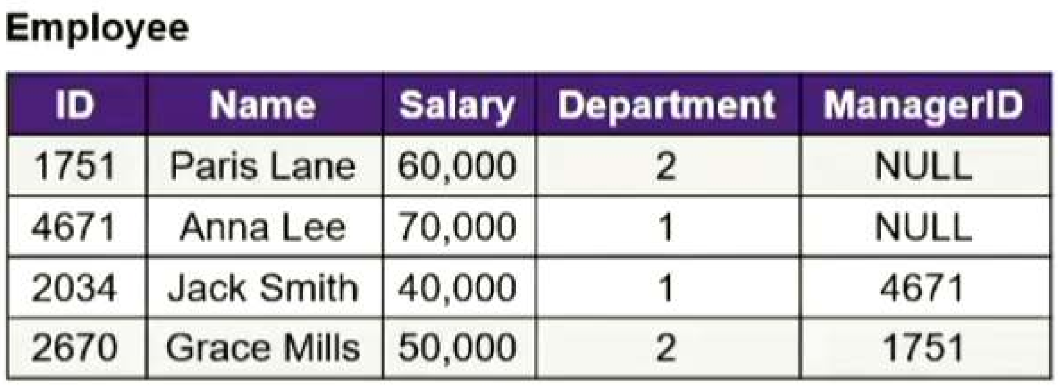
Department

ID	Name	Manager
1	Marketing	4671
2	Development	1751

Employee

ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1

Domains for Department.ID and Employee.ID are different, even though the attribute names are the same



Employee

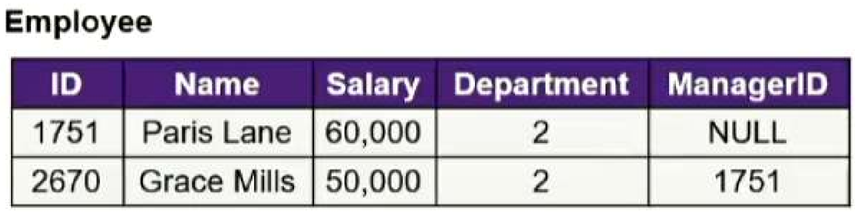
ID	Name	Salary	Department	ManagerID
1751	Paris Lane	60,000	2	NULL
4671	Anna Lee	70,000	1	NULL
2034	Jack Smith	40,000	1	4671
2670	Grace Mills	50,000	2	1751

Domains for ID and ManagerID are the same but the attribute names are different

Tuple t is an ordered list of values

$$t = <v_1, v_2, \dots, v_n>$$

where each value v_i ($1 \leq i \leq n$) is an element of the corresponding domain of attribute A_i or a special value called "NULL"



Employee

ID	Name	Salary	Department	ManagerID
1751	Paris Lane	60,000	2	NULL
2670	Grace Mills	50,000	2	1751

Relationship Schema
Denoted by R[A1, A2, A3, …, An], includes a relation name R and list of attributes.
Integer n is termed "degree of the relation"
A relation schema of degree 5

- Employee[id, name, sex, salary, department]

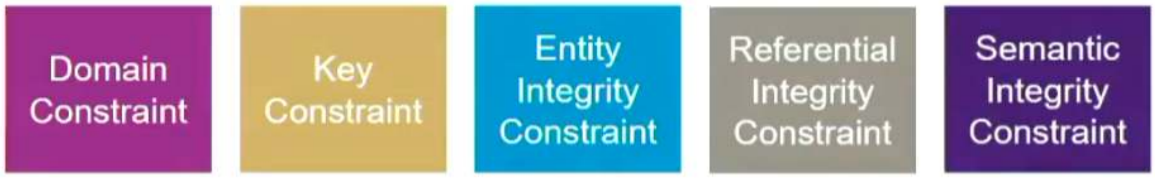
Relation Instance

- A relation instance r of the relation schema R, denoted by r(R), is a set of n-tuples r = {t1, t2, …, tm}.

Ordering of values within A tuple
N-tuple is an ordered list of n values
Syntactically, all tuples in relation have values in the same order

Semantically, the order chosen is irrelevant, as long as the correspondence between the attributes and the values can be maintained.

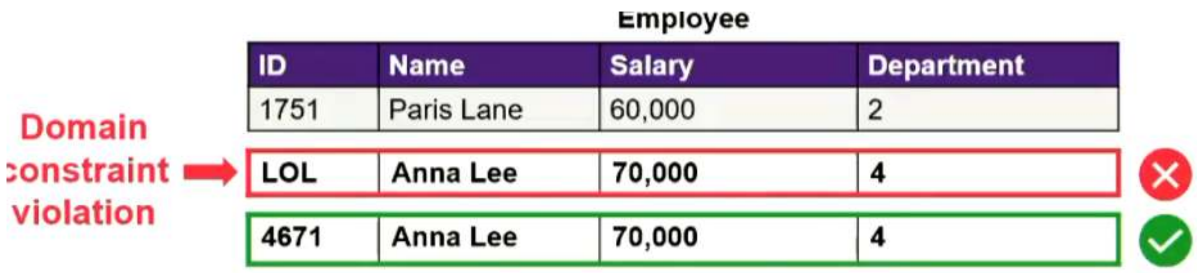
Database Integrity Constraints
Are rules that enforce the 'integrity', or the correctness, of our database.
They must hold on every instance of the schema.



Domain Constraint | Key Constraint | Entity Integrity Constraint | Referential Integrity Constraint | Semantic Integrity Constraint

D E K R S

Domain Constraints:
A domain constraint violation occurs when an attribute's value does not appear in the corresponding domain.



Domain constraint violation

Employee

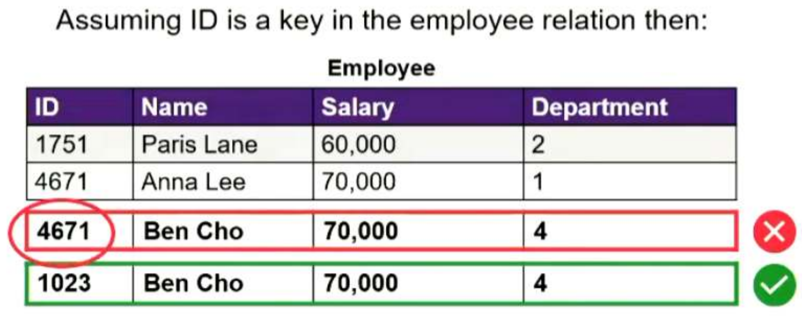
ID	Name	Salary	Department	
1751	Paris Lane	60,000	2	
LOL	**Anna Lee**	**70,000**	**4**	✗
4671	**Anna Lee**	**70,000**	**4**	✓

Key Constraints:
A key constraint violation occurs when a tuple is inserted or modified such that it has the same key value as another tuple.

A key is a minimal set of attributes that uniquely identify tuples in a relation. The term minimal does not mean the smallest set of attributes but instead a set of attributes without any redundant attributes.

A schema may have more than one key.

- Each is called a candidate key
- One is selected as the primary key, which would be underlined.



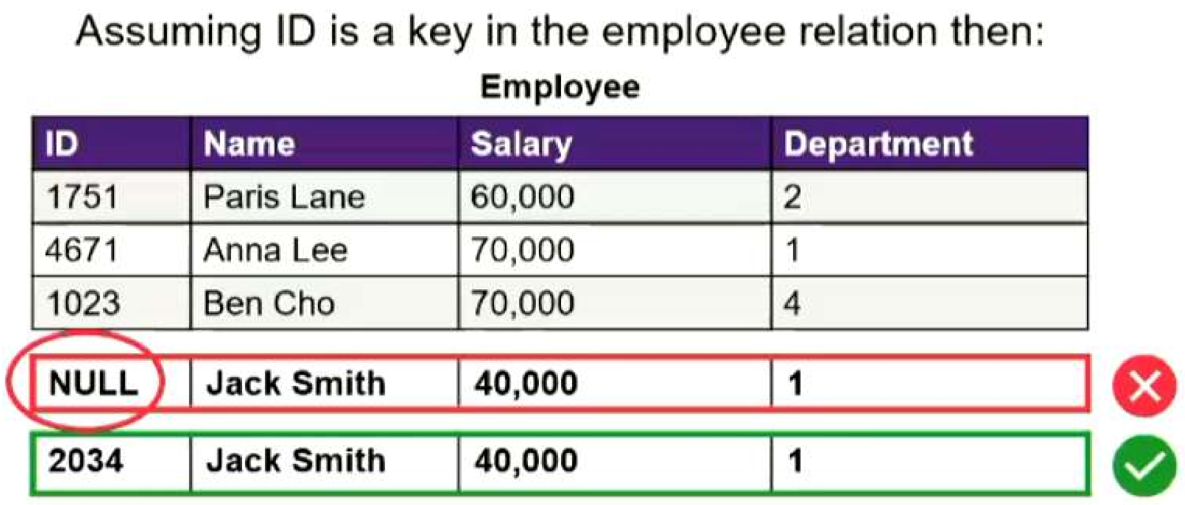
Assuming ID is a key in the employee relation then:

Employee

ID	Name	Salary	Department	
1751	Paris Lane	60,000	2	
4671	Anna Lee	70,000	1	
4671	**Ben Cho**	**70,000**	**4**	✗
1023	**Ben Cho**	**70,000**	**4**	✓

An entity key constraint violation occurs when a tuple is inserted or modified such that part of its primary key contains the value NULL.
For primary keys that consist of multiple attributes, no part of the primary key can be null.

Assuming ID is a key in the employee relation then:



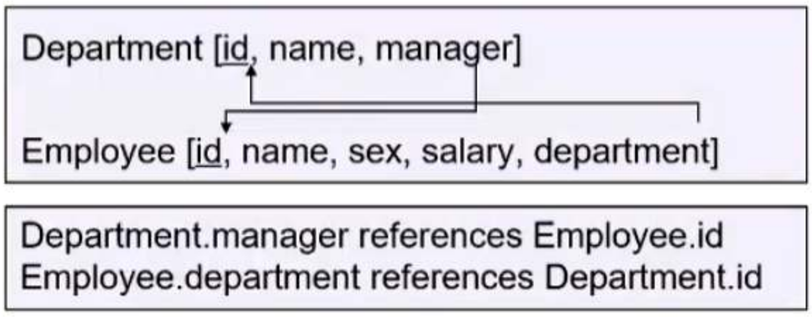
Assuming ID is a key in the employee relation then:

Employee

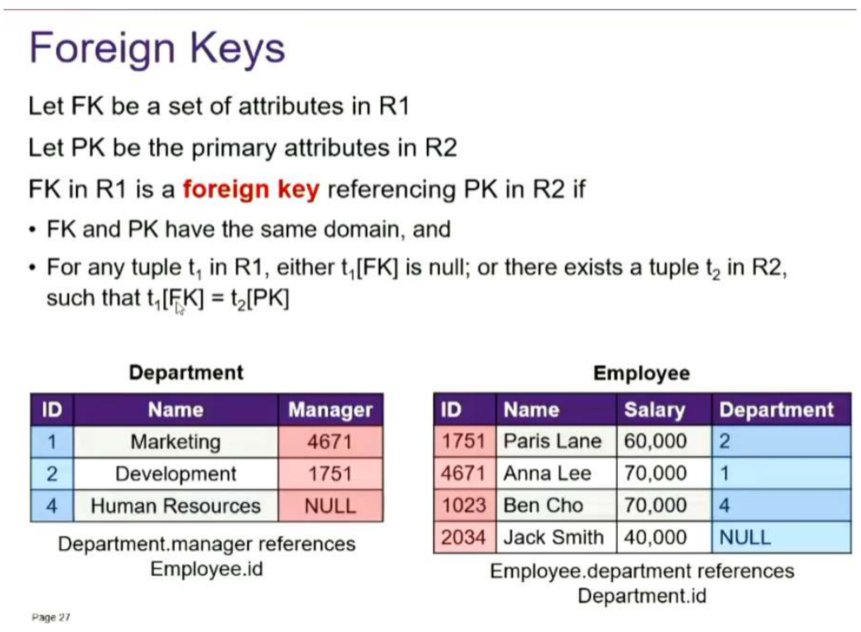
ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1
1023	Ben Cho	70,000	4
NULL	**Jack Smith**	**40,000**	**1**
2034	**Jack Smith**	**40,000**	**1**

A referential integrity constraint violation occurs when any operation performed on our database leads to an invalid, incorrect or broken foreign key reference.

Foreign keys allow us to relate two different schemas. This can be viewed graphically or textually.


Department [id, name, manager]
Employee [id, name, sex, salary, department]

Department.manager references Employee.id
Employee.department references Department.id



Foreign Keys

Let FK be a set of attributes in R1

Let PK be the primary attributes in R2

FK in R1 is a **foreign key** referencing PK in R2 if

- FK and PK have the same domain, and
- For any tuple t_1 in R1, either t_1[FK] is null; or there exists a tuple t_2 in R2, such that t_1[FK] = t_2[PK]

Department

ID	Name	Manager
1	Marketing	4671
2	Development	1751
4	Human Resources	NULL

Department.manager references Employee.id

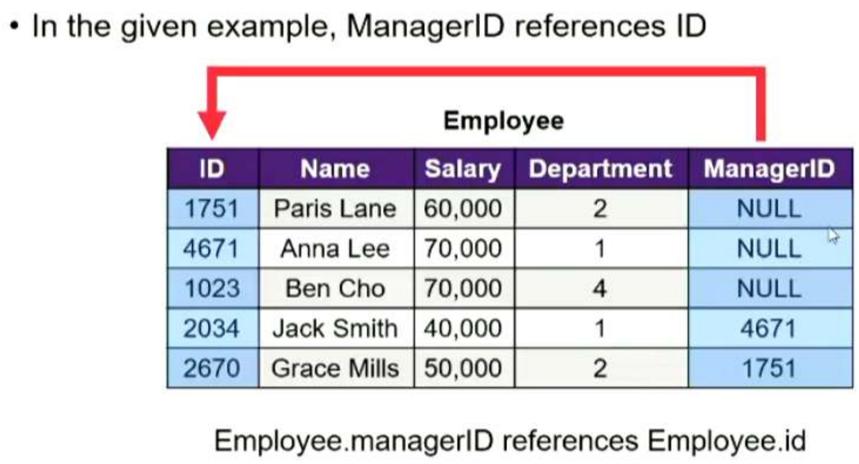
Employee

ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1
1023	Ben Cho	70,000	4
2034	Jack Smith	40,000	NULL

Employee.department references Department.id

Page 27

Self-referencing Relations
It is also possible for a table to reference itself.



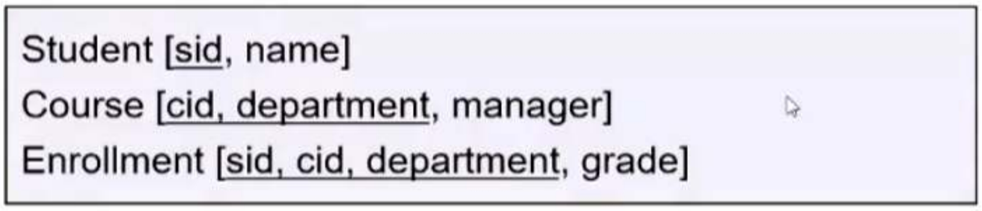
- In the given example, ManagerID references ID

Employee

ID	Name	Salary	Department	ManagerID
1751	Paris Lane	60,000	2	NULL
4671	Anna Lee	70,000	1	NULL
1023	Ben Cho	70,000	4	NULL
2034	Jack Smith	40,000	1	4671
2670	Grace Mills	50,000	2	1751

Employee.managerID references Employee.id

Relations with Composite keys
Also possible to have FKs to relations that have a multi-attribute primary key.



Student [sid, name]
Course [cid, department, manager]
Enrollment [sid, cid, department, grade]

Enrollment.sid references Student.sid
Enrollment.{cid, department} references Course.{cid, department}

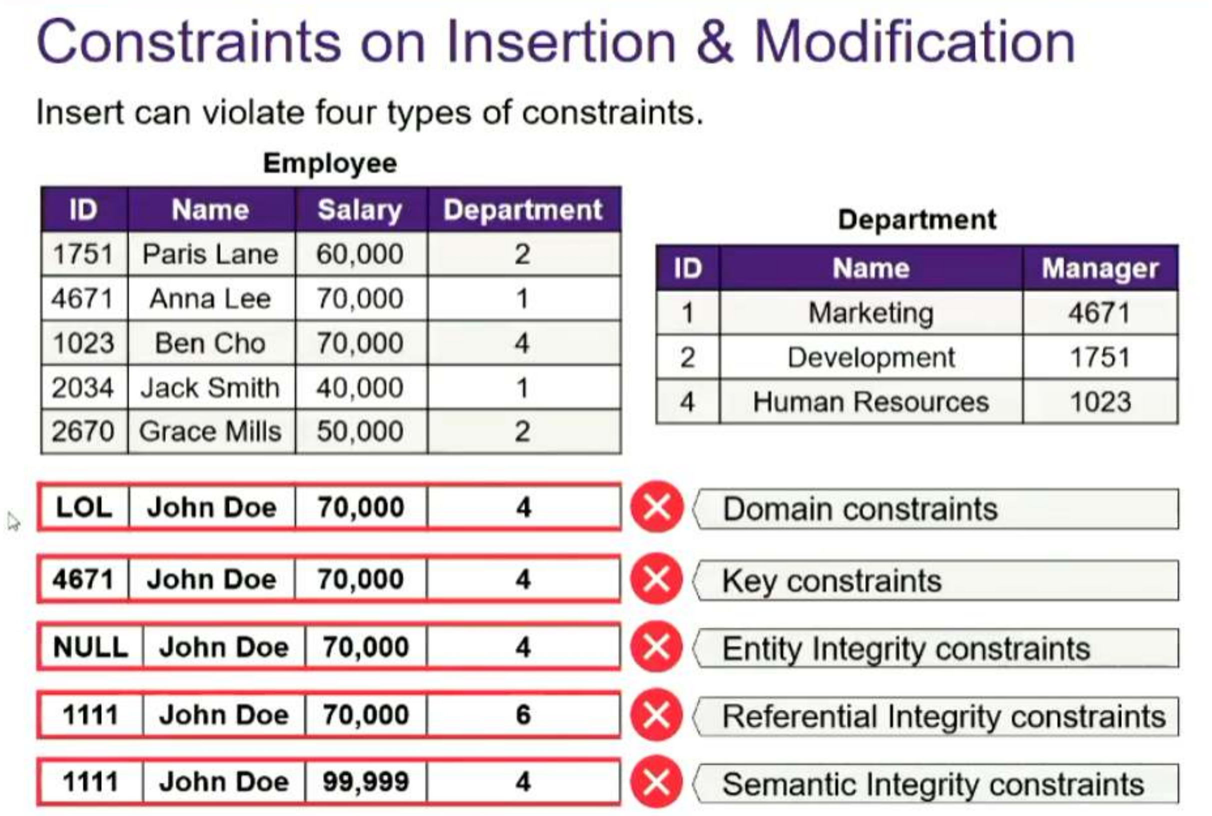
Semantic Integrity Constraint
Are generally defined by the business or organisation during client consultation.

Semantic constraints can be used to enforce organisations policies such as:
- The salary of an employee should not exceed the employee's supervisor's salary"
- The maximum number of hours that an employee can work on a project is 56.

Constraints and Operations
Enforcement of integrity constraints ensures that the database remains consistent.

Changes to the database such as insert, medication and deletion musn't violate integrity constraints (leave the database in al inconsistent state)

If an update is submitted that would violate the integrity, it must be rejected.



Constraints on Insertion & Modification

Insert can violate four types of constraints.

Employee

ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1
1023	Ben Cho	70,000	4
2034	Jack Smith	40,000	1
2670	Grace Mills	50,000	2

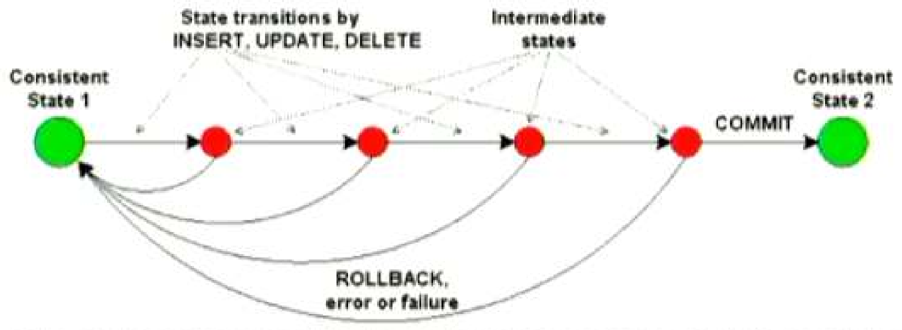
Department

ID	Name	Manager
1	Marketing	4671
2	Development	1751
4	Human Resources	1023

LOL	John Doe	70,000	4	✗	Domain constraints
4671	John Doe	70,000	4	✗	Key constraints
NULL	John Doe	70,000	4	✗	Entity Integrity constraints
1111	John Doe	70,000	6	✗	Referential Integrity constraints
1111	John Doe	99,999	4	✗	Semantic Integrity constraints

Deletion operations can lead to referretnail or semantic integrity constraints.
These violations can either be rejected, cascade or the referencing attribute values can be modified to a default value.

A transaction is an executing program that includes some database operations, such as reading from the database, or applying insertions, deletions, or updates to the database.
At the end of the transaction, it must leave the database in a valid or consistante state that satisfies all the constraints specified on the database schema.

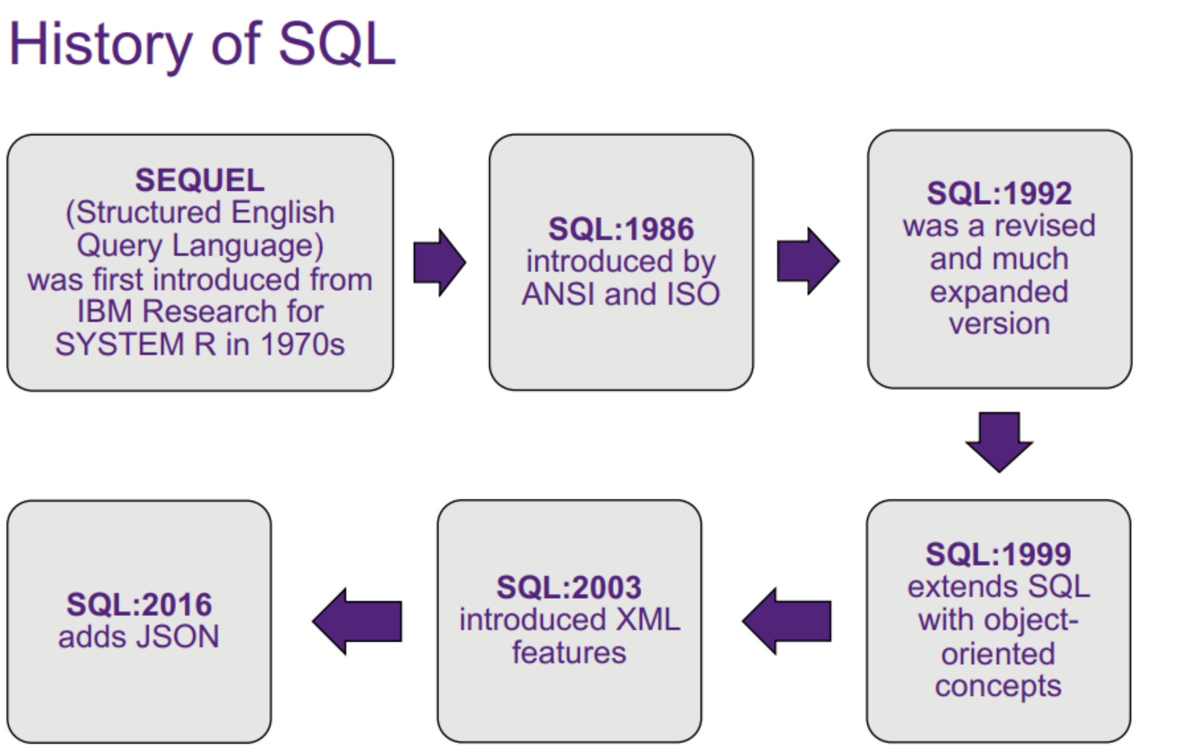

State transitions by INSERT, UPDATE, DELETE
Intermediate states
Consistent State 1
Consistent State 2
COMMIT
ROLLBACK, error or failure

Source: https://maxdb.sap.com/doc/7 7/81/74b30edc2142658e510080ef6917f1/ppt img.gif

Transactions allow execution of a suite of queries where constraint violations in intermediate steps are allowed.

Week 6

- **Relational Algebra (RA)**
  - **Formal query language for relational database.**
- **Structured Query Language (SQL)**
  - Implementation of RA
  - Comprehensive, commercial query language with widely accepted international standard.
- **Query by Example (QBE)**
  - Commercial, graphical query language with minimum syntax.


History of SQL
SEQUEL (Structured English Query Language) was first introduced from IBM Research for SYSTEM R in 1970s
SQL:1986 introduced by ANSI and ISO
SQL:1992 was a revised and much expanded version
SQL:1999 extends SQL with object-oriented concepts
SQL:2003 introduced XML features
SQL:2016 adds JSON

Three types of SQL statements
Data Definition Language

- Statements to define the database schema

Data Manipulation Language (DML)

- Statements to manipulate the data

Data Control Language (DCL)

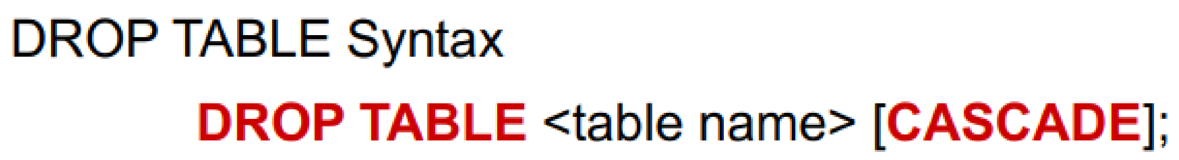
- Statements to specify transaction control, semantic integrity (triggers and assertions), authorization and management of privileges
- Statements for specifying the physical storage parameters such as the file structures and access paths (indexes)
- Statements to specify the role-based security controls

Basic sql DDL statements
Table Definition

1. Drop Table
2. Create Table
3. Alter Table

Drop Table

- Drops all constraints defined on the table including constraints in other tables which reference this table.
- Deletes all tuples within the table
- Removes the table definition from the system catalogue.



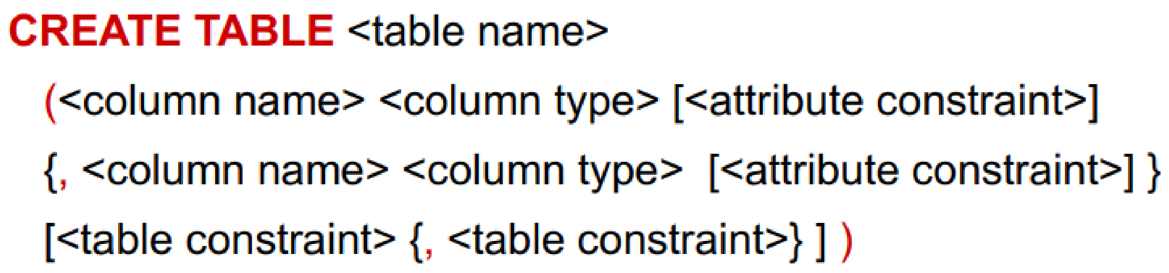
DROP TABLE Syntax

DROP TABLE <table name> **[CASCADE]**;

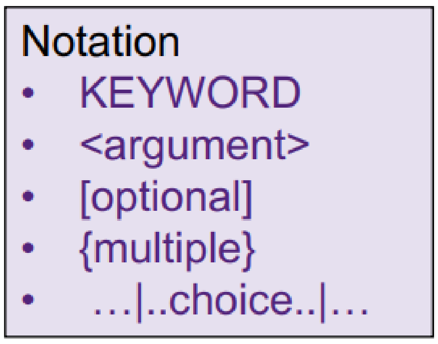
Creating Tables
Creates a new relation, by specifying its name, attributes and constraints.
The key, entity and referential integrity constraints are specified within the statement after the attributes have been declared.

The domain constraint is specified for each attribute

Data type of an attribute can be specified directly or by declaring a domain (CREATE DOMAIN).



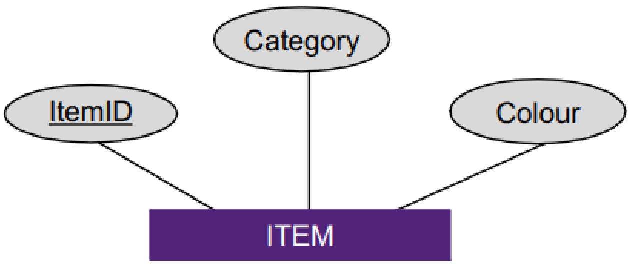
```
CREATE TABLE <table name>
  (<column name> <column type> [<attribute constraint>]
  {, <column name> <column type>  [<attribute constraint>] }
  [<table constraint> {, <table constraint>} ] )
```



Notation

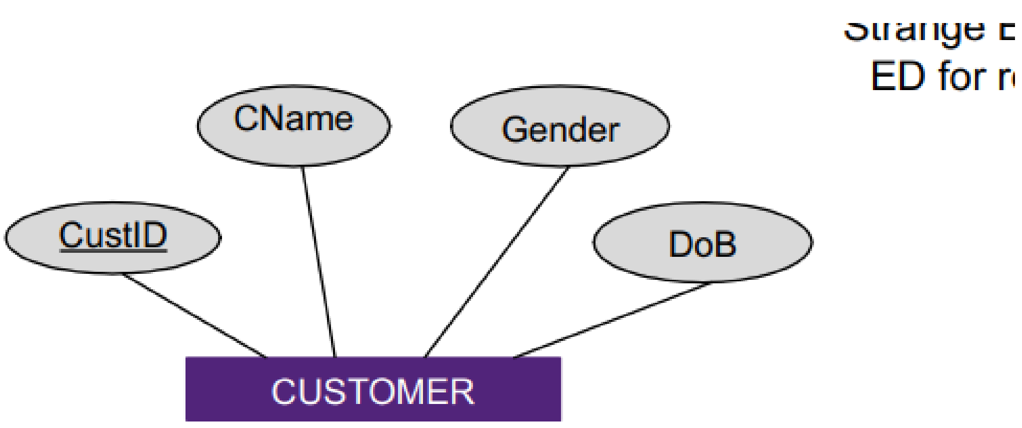
- KEYWORD
- <argument>
- [optional]
- {multiple}
- …|..choice..|…

Remember to
database t
Hint: Left Colun



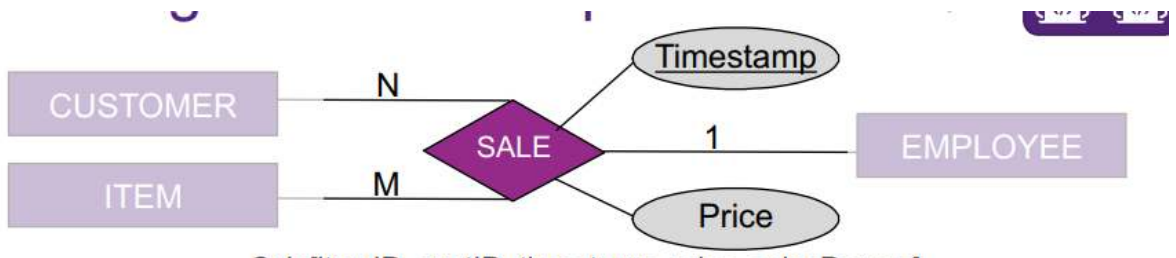
Item [itemID, category, color]

```
CREATE TABLE Item (
          itemID     INTEGER,
          category   ENUM('food', 'clothing', 'furniture'),
          colour     CHAR(3),
          PRIMARY KEY (itemID));
```



Customer [custID, cname, gender, dob]

```
CREATE TABLE Customer (
        custID      INTEGER,
        cname       CHAR(20),
        gender      CHAR(10),
        dob         DATE,
        PRIMARY KEY (custID));
```



Sale[itemID, custID, timestamp, price, salesPerson]
Sale.itemID references Item.itemID
Sale.custID references Customer.custID
Sale.salesperson references Employee.ssn

```
CREATE TABLE Sale (
  itemID            INTEGER,
  custID            INTEGER,
  timestamp         TIMESTAMP,
  price             DOUBLE(8, 2),
  salesPerson       CHAR(9),
  PRIMARY KEY (itemID, custID, timestamp),
  FOREIGN KEY (itemID)  REFERENCES Item(itemID),
  FOREIGN KEY (custID)  REFERENCES Customer(custID),
  FOREIGN KEY (salesPerson)  REFERENCES Employee(ssn));
```

The purpose of **constraints** is to **limit** what data can go into a table.
Constraints act like **rules** for a table. Let's consider some examples of CONSTRAINTS you are already familiar with,

- **PRIMARY KEY:** Ensures attribute value is unique and not null.
- **FOREIGN KEY:** Ensures attribute value exists in parent table.

and some new examples.

- **CHECK:** Ensures attribute values meets a predefined condition(s)
- **UNIQUE:** Ensures attribute value is unique or null.

You can give **constraints** a **name** which has the following benefits:
1. Easier to understand errors. If a query violates a constraint, SQL will generate an error message that will contain the constraint name.
2. Easier to modify or remove the constraint.
- E.g. ALTER TABLE Item DROP CONSTRAINT item_pk;

Semantic constraints over a single table are specified using Check conditional-expressions

```
CREATE TABLE Sale (
            itemID          INTEGER,
            custID          INTEGER,
            timestamp       TIMESTAMP,
            price           DOUBLE(8, 2),
            salesPerson     CHAR(9),

            PRIMARY KEY (itemID, custID, timestamp),
            FOREIGN KEY (itemID)  REFERENCES Item(itemID),
            FOREIGN KEY (custID)  REFERENCES Customer(custID),
            FOREIGN KEY (salesPerson)  REFERENCES Employee(ssn),
            CHECK (price >= 8.50 AND price < 150000)
);
```

Alter table command is used for schema evolution, that is, the definition of a table created using the CREATE TABLE command can be changed using the ALTER TABLE command.

```
ALTER TABLE <table name>
  ADD <column name> <column type> [<attribute constraint>]
      {, <column name> <column type> [<attribute constraint>] }
  | DROP <column name> [CASCADE]
  | MODIFY <column name> <column-options>
  | ADD <constraint name> <constraint-options>
  | DROP <constraint name> [CASCADE];
```

To alter a constraint, it must be dropped and added again.

Note: Commercial products have variations!

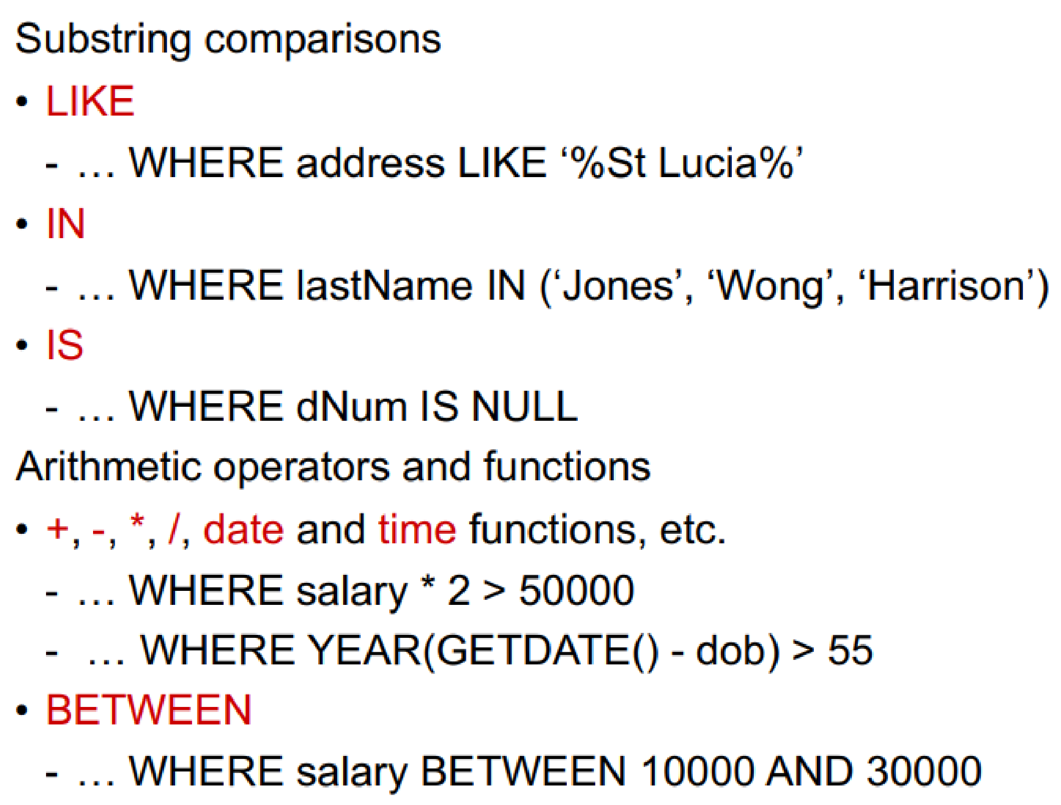
Notation
- KEYWORD
- <argument>
- [optional]
- {multiple}
- …|..choice..|…

Data Manipulation Language (DML)

DML statements are used for managing data within schema objects.
• **INSERT** – insert data into a table.
• **UPDATE** – updates existing data within a table.
• **DELETE** – deletes records from a table.
• **SELECT** – retrieve data from a database

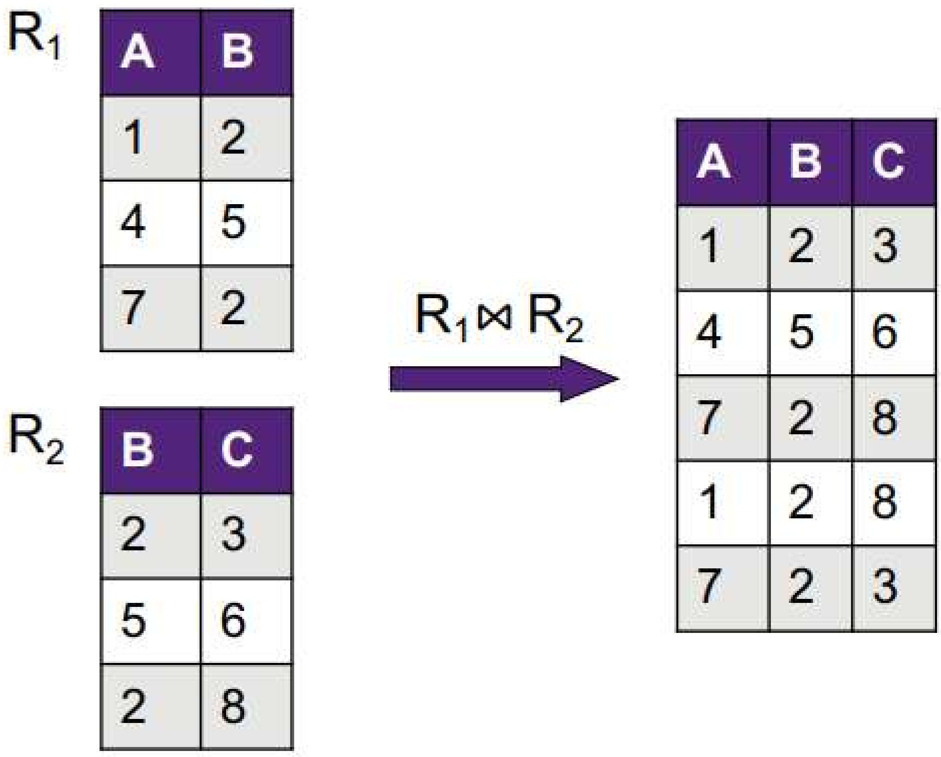
Insert is used to add tuples to an existing relation
DELETE statement is used to remove existing tuples from a relation
UPDATE statement is used to modify attribute values of one or more selected tuples in a relation

The SELECT statement, users specify what the result of the query should be, and the DBMS decides the operations and order of execution, thus SQL queries are declarative


Substring comparisons
• LIKE
  - … WHERE address LIKE '%St Lucia%'
• IN
  - … WHERE lastName IN ('Jones', 'Wong', 'Harrison')
• IS
  - … WHERE dNum IS NULL

Arithmetic operators and functions
• +, -, *, /, date and time functions, etc.
  - … WHERE salary * 2 > 50000
  - … WHERE YEAR(GETDATE() - dob) > 55
• BETWEEN
  - … WHERE salary BETWEEN 10000 AND 30000

Equi-Join
Joins tuples from two relations when they have the same value for some pair(s) of designated attributes. The specification is termed a "join condition"
With equi-join, the join condition only has equality comparisons



R_1

A	B
1	2
4	5
7	2

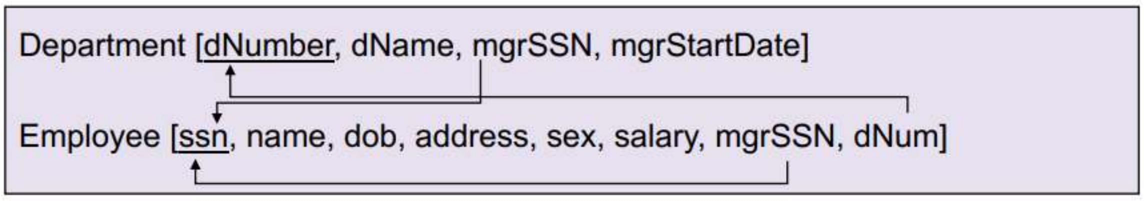
R_2

B	C
2	3
5	6
2	8

$R_1 \bowtie R_2$

A	B	C
1	2	3
4	5	6
7	2	8
1	2	8
7	2	3

A join is used to combine related tuples from two relations into a single tuple in a new (result) relation
Join operation is needed for organising a search space of data. This is needed when information is contained in more than one relation



Department [dNumber, dName, mgrSSN, mgrStartDate]

Employee [ssn, name, dob, address, sex, salary, mgrSSN, dNum]

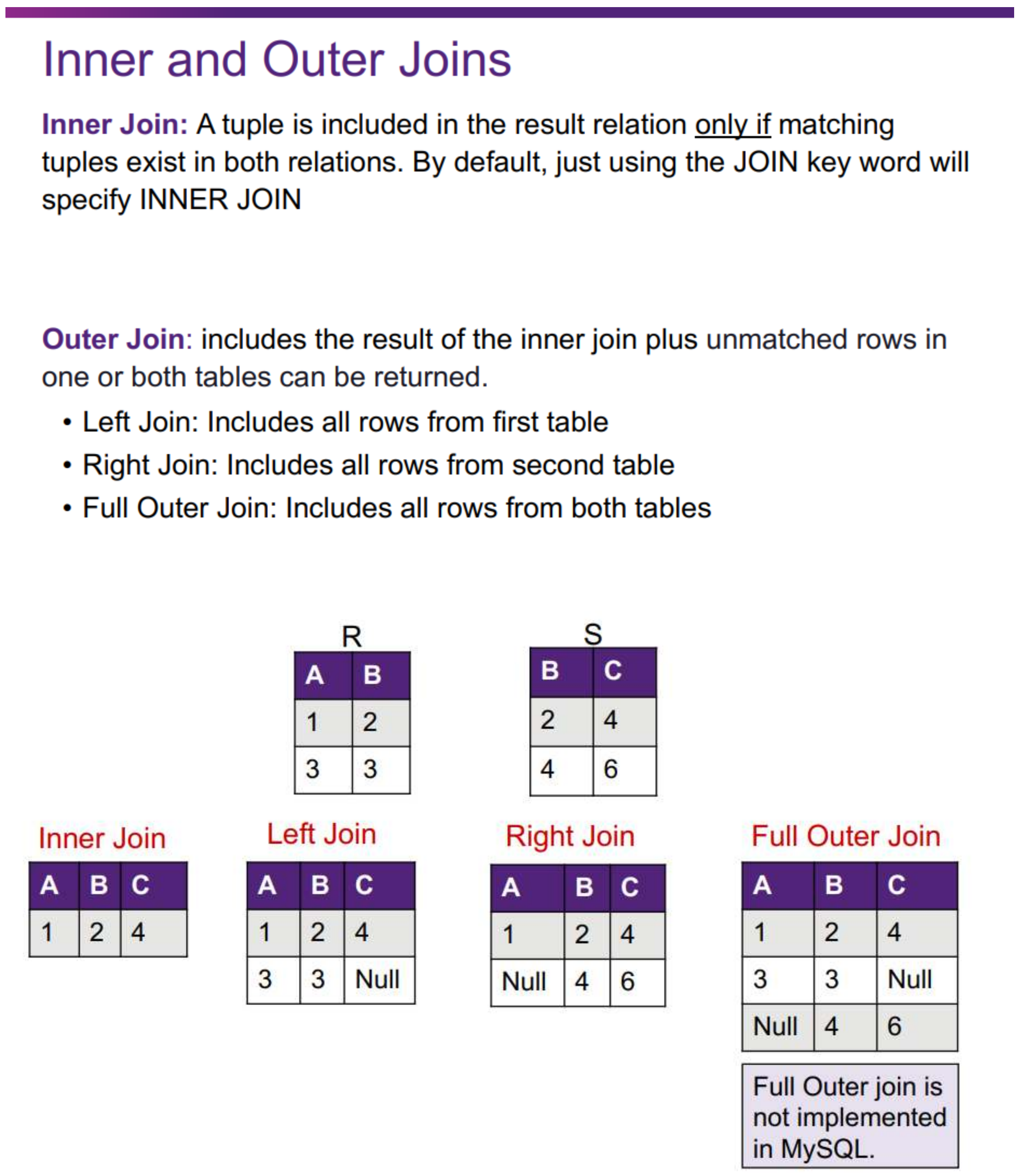
1. Equi-Join using **WHERE**

```
SELECT    E.name, D.dName
FROM      Department AS D, Employee AS E
WHERE     D.mgrSSN = E.ssn;
```

2. Equi-Join using **JOIN**

```
SELECT    E.name, D.dName
FROM      Department AS D
JOIN      Employee AS E
ON        D.mgrSSN = E.ssn
```

Theta-Join
A join operation is used to select a subset of the Cartesian Product
The most general type of join is called theta-join
Join condition allows several logical operators {=, ^{1}, <, £, >, ^{3}}



Inner and Outer Joins

Inner Join: A tuple is included in the result relation only if matching tuples exist in both relations. By default, just using the JOIN key word will specify INNER JOIN

Outer Join: includes the result of the inner join plus unmatched rows in one or both tables can be returned.

- Left Join: Includes all rows from first table
- Right Join: Includes all rows from second table
- Full Outer Join: Includes all rows from both tables

R

A	B
1	2
3	3

S

B	C
2	4
4	6

Inner Join

A	B	C
1	2	4

Left Join

A	B	C
1	2	4
3	3	Null

Right Join

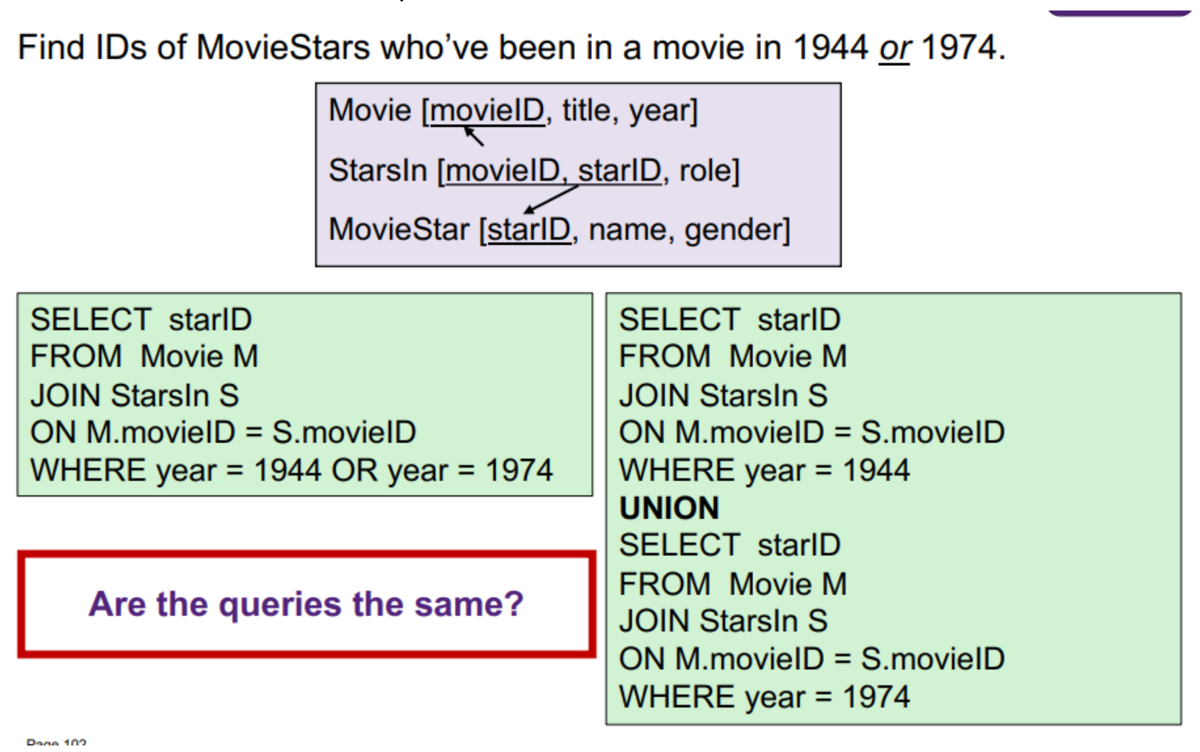
A	B	C
1	2	4
Null	4	6

Full Outer Join

A	B	C
1	2	4
3	3	Null
Null	4	6

Full Outer join is not implemented in MySQL.

UNION: Produces a relation that includes all tuples that appear only in R1, or only in R2, or in both R1 and R2.
• Duplicate tuples are eliminated if UNION ALL is not used.
• R1 and R2 must be union compatible



Find IDs of MovieStars who've been in a movie in 1944 *or* 1974.

Movie [movieID, title, year]

StarsIn [movieID, starID, role]

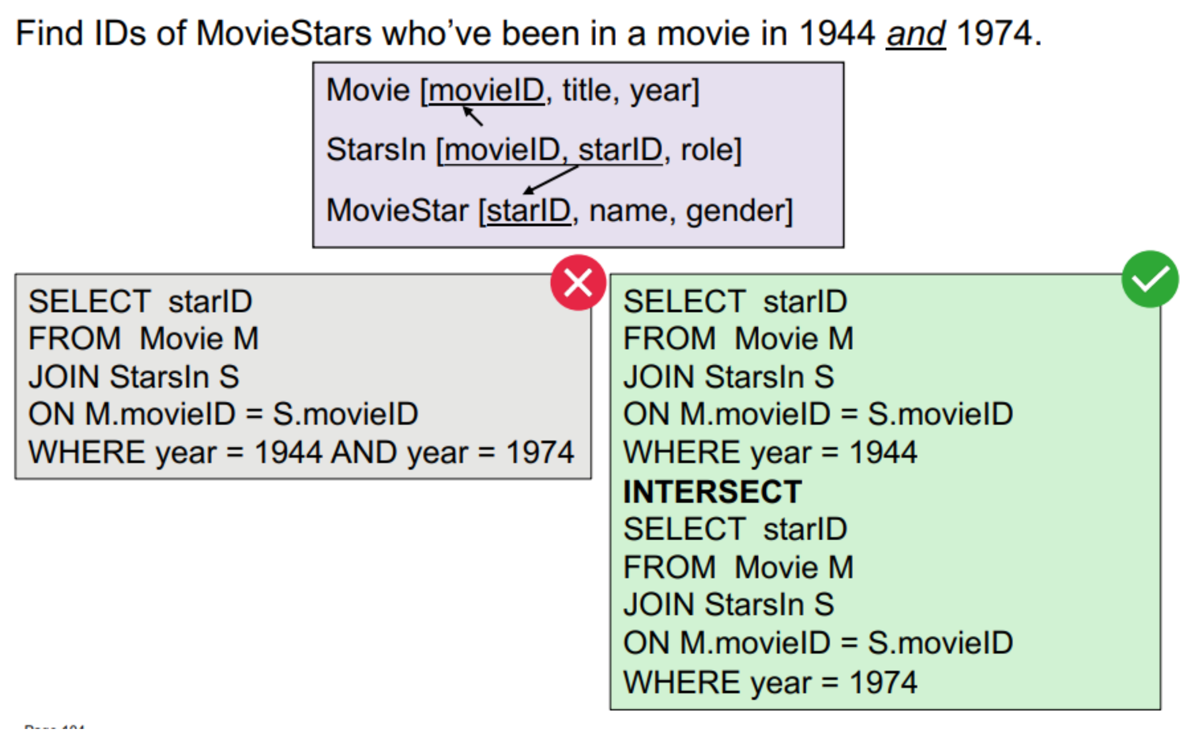
MovieStar [starID, name, gender]

```
SELECT  starID
FROM  Movie M
JOIN StarsIn S
ON M.movieID = S.movieID
WHERE year = 1944 OR year = 1974
```

Are the queries the same?

```
SELECT  starID
FROM  Movie M
JOIN StarsIn S
ON M.movieID = S.movieID
WHERE year = 1944
UNION
SELECT  starID
FROM  Movie M
JOIN StarsIn S
ON M.movieID = S.movieID
WHERE year = 1974
```

INTERSECTION: Produces a relation that includes the tuples that appear in both R1 and R2.
• Duplicate tuples are eliminated if INTERSECT ALL is not used.
• R1 and R2 must be union compatible.



Find IDs of MovieStars who've been in a movie in 1944 *and* 1974.

Movie [movieID, title, year]

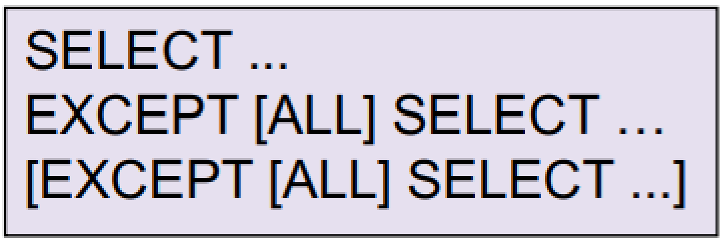
StarsIn [movieID, starID, role]

MovieStar [starID, name, gender]

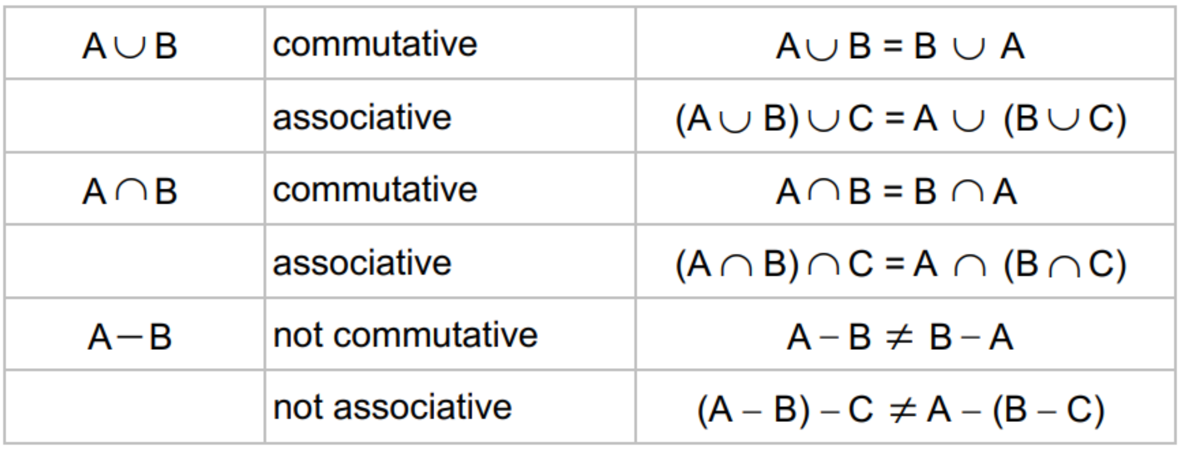
```
SELECT  starID
FROM  Movie M
JOIN StarsIn S
ON M.movieID = S.movieID
WHERE year = 1944 AND year = 1974
```

```
SELECT  starID
FROM  Movie M
JOIN StarsIn S
ON M.movieID = S.movieID
WHERE year = 1944
INTERSECT
SELECT  starID
FROM  Movie M
JOIN StarsIn S
ON M.movieID = S.movieID
WHERE year = 1974
```

EXCEPT (also referred to as MINUS) produces a relation that includes all the tuples that appear in R1, but do not appear in R2. • R1 and R2 must be union compatible.



```
SELECT ...
EXCEPT [ALL] SELECT …
[EXCEPT [ALL] SELECT ...]
```



$A \cup B$	commutative	$A \cup B = B \cup A$
	associative	$(A \cup B) \cup C = A \cup (B \cup C)$
$A \cap B$	commutative	$A \cap B = B \cap A$
	associative	$(A \cap B) \cap C = A \cap (B \cap C)$
$A - B$	not commutative	$A - B \neq B - A$
	not associative	$(A - B) - C \neq A - (B - C)$

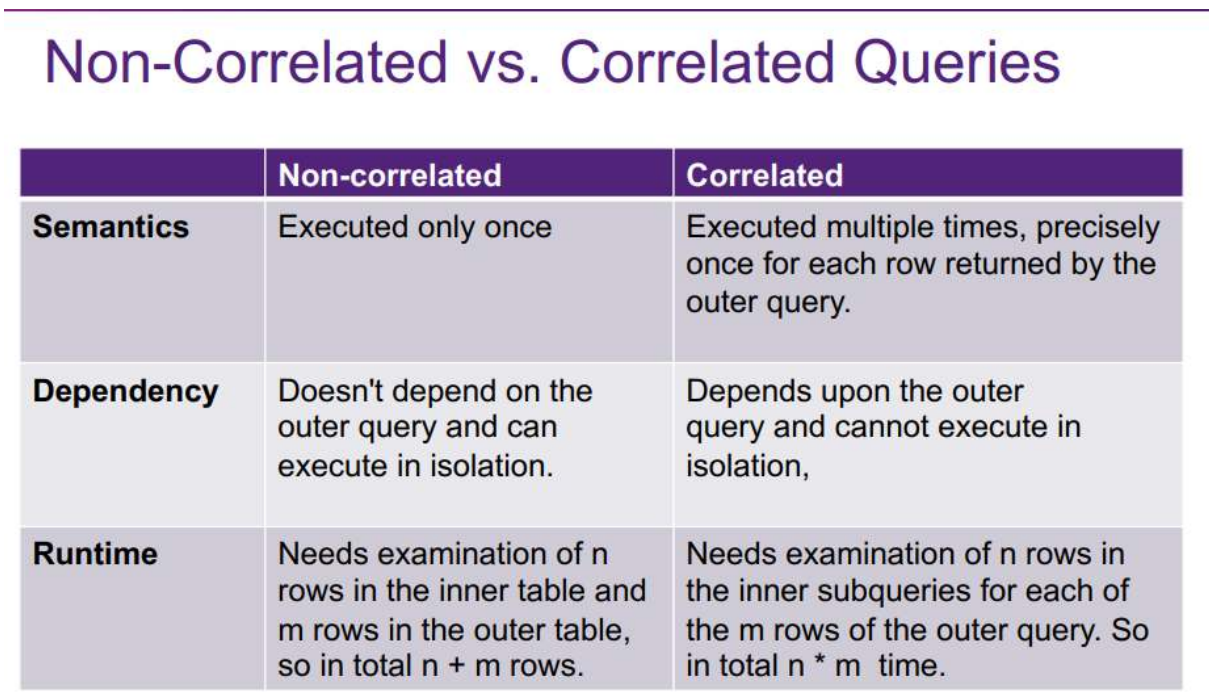
Nested SQL Queries
A nested query (often termed subquery) is a query that appears within another query.
Inside the WHERE clause of another SELECT statement
Inside an INSERT, UPDATE or DELETE statement
Nesting can occur at multiple levels
The equity that contains the nested query is called its outer query.

Two types of Subqueries
Non-correlated

- Results are returned from an inner equity to an outer clause
- Subqueries are evaluated from the “inside out”
- The outer query takes an action based on the results of the inner query

Correlated

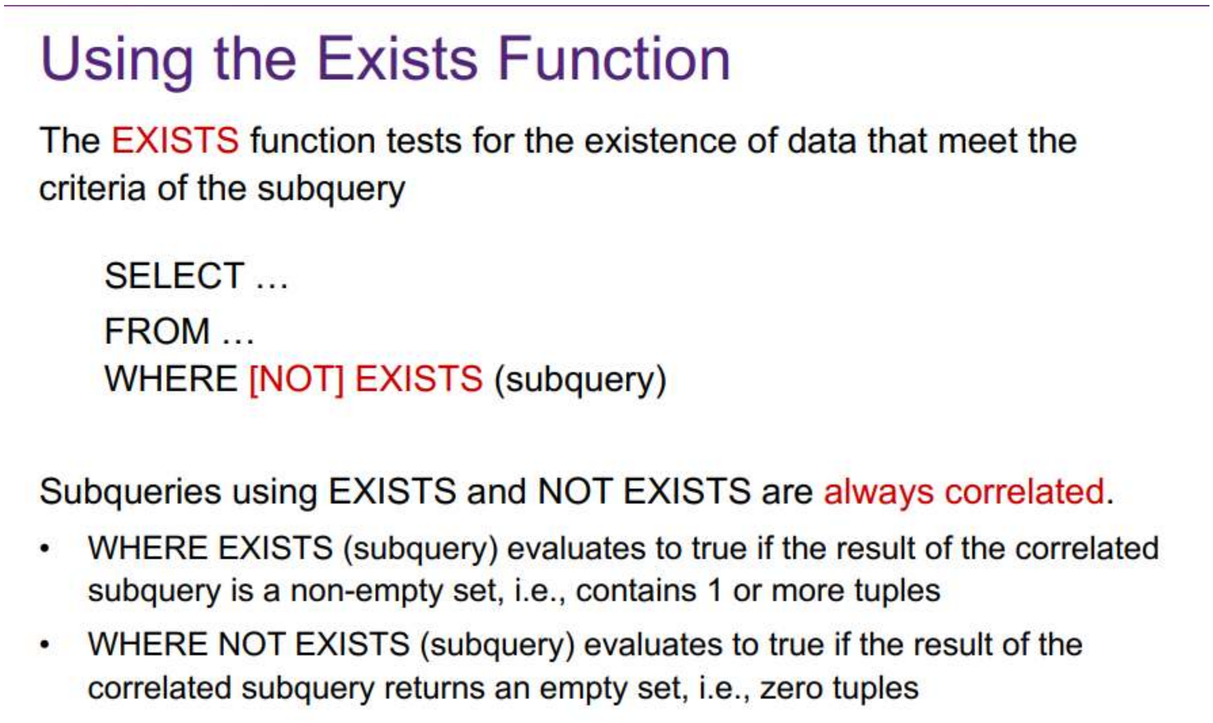
- Conditions in subquery WHERE clause have references to some attributes of a relation in the outer query
- The outer SQL statement provides the values for the inner subquery to use in its evaluation
- The subquery is evaluated once for each (combination of) tuple in the outer query



Non-Correlated vs. Correlated Queries

	Non-correlated	Correlated
Semantics	Executed only once	Executed multiple times, precisely once for each row returned by the outer query.
Dependency	Doesn't depend on the outer query and can execute in isolation.	Depends upon the outer query and cannot execute in isolation,
Runtime	Needs examination of n rows in the inner table and m rows in the outer table, so in total n + m rows.	Needs examination of n rows in the inner subqueries for each of the m rows of the outer query. So in total n * m time.

Exists Function



Using the Exists Function

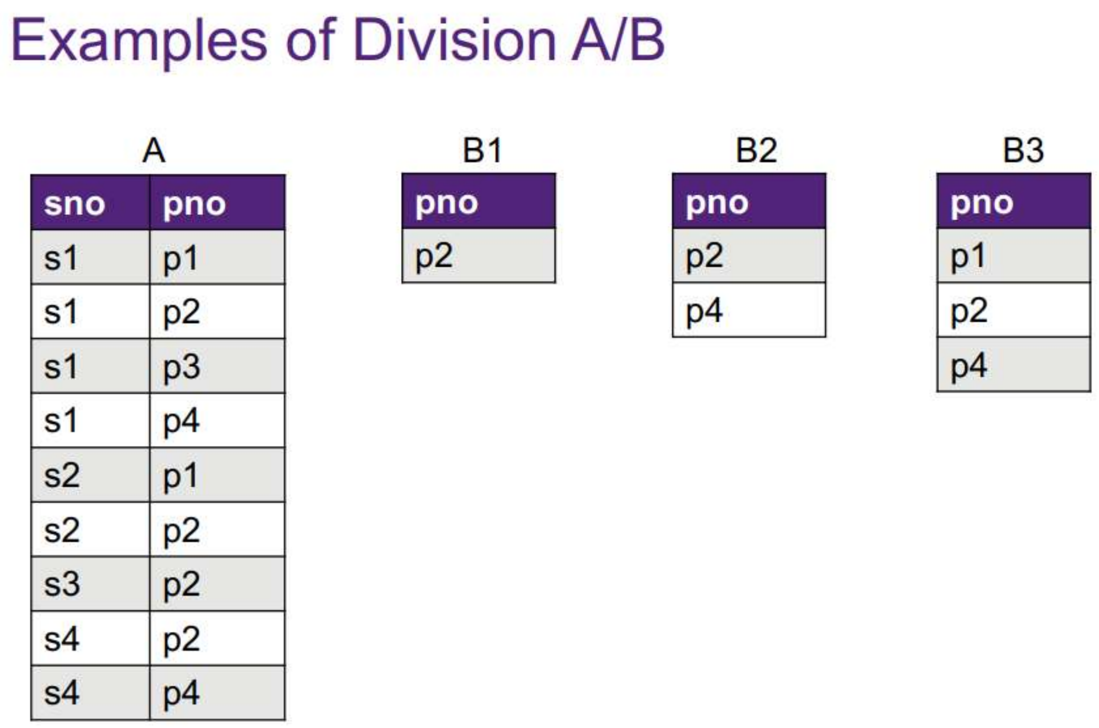
The EXISTS function tests for the existence of data that meet the criteria of the subquery

```
SELECT …
FROM …
WHERE [NOT] EXISTS (subquery)
```

Subqueries using EXISTS and NOT EXISTS are always correlated.

- WHERE EXISTS (subquery) evaluates to true if the result of the correlated subquery is a non-empty set, i.e., contains 1 or more tuples
- WHERE NOT EXISTS (subquery) evaluates to true if the result of the correlated subquery returns an empty set, i.e., zero tuples

Division
Division is useful for answering queries which include a “for all” or “for every” phrase, e.g., find movie stars who were in all movies.
Dividing R1/R2
• Resulting relation contains columns in R1, but not in R2
• Relations R1 and R2 must be division compatible, i.e. (last) n columns of R1 must be identically named to columns in R2, where n is the degree of R2
• The resulting relation contains tuples t, such that a value in t appears in R1, in combination with every tuple in R2



Examples of Division A/B

A

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B1

pno
p2

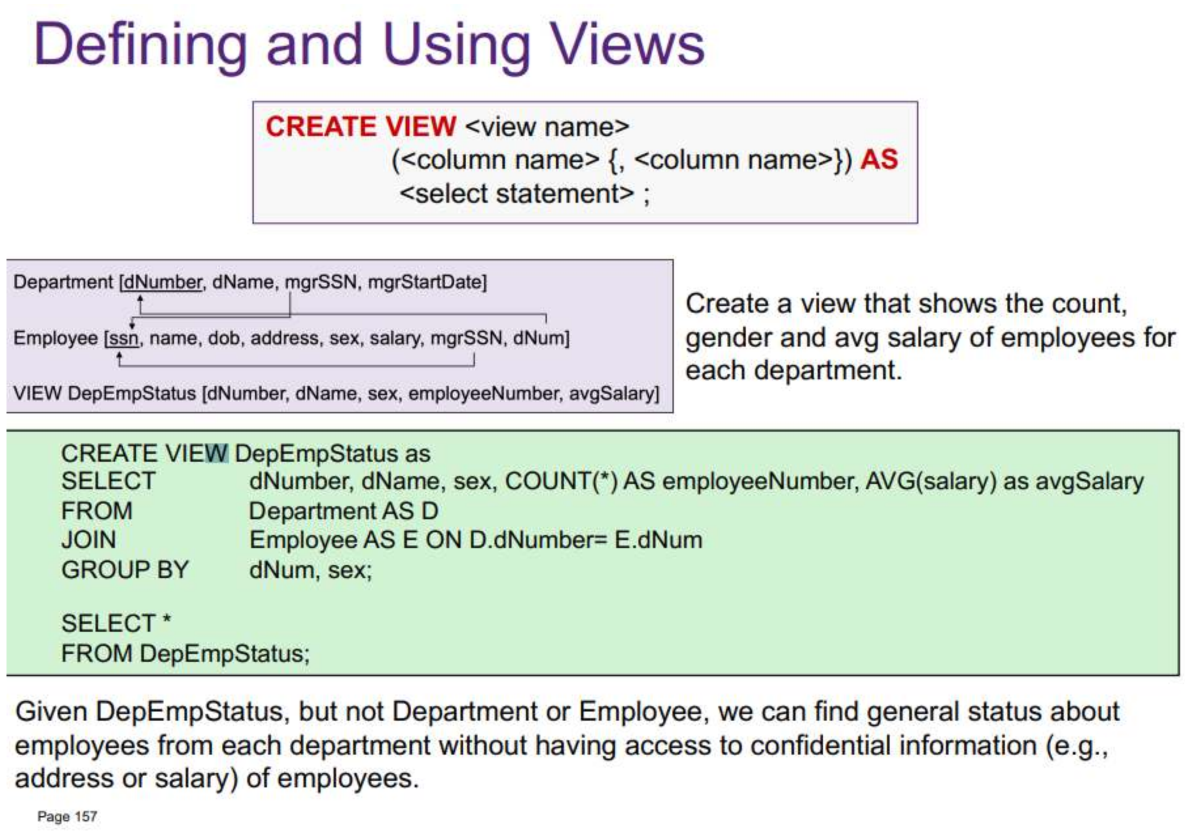
B2

pno
p2
p4

B3

pno
p1
p2
p4

Views
A view is a single table that is derived from other tables, which could be base tables or previously defined views
Views can be
• Virtual tables - that do not physically exist on disk.
• Materialised - by physically creating the view table. These must be updated when the base tables are updated
We can think of a virtual view as a way of specifying a table that we need to reference frequently, even though it does not physically exist.



Defining and Using Views

```
CREATE VIEW <view name>
            (<column name> {, <column name>}) AS
             <select statement> ;
```

Department [dNumber, dName, mgrSSN, mgrStartDate]

Employee [ssn, name, dob, address, sex, salary, mgrSSN, dNum]

VIEW DepEmpStatus [dNumber, dName, sex, employeeNumber, avgSalary]

Create a view that shows the count, gender and avg salary of employees for each department.

```
CREATE VIEW DepEmpStatus as
SELECT          dNumber, dName, sex, COUNT(*) AS employeeNumber, AVG(salary) as avgSalary
FROM            Department AS D
JOIN            Employee AS E ON D.dNumber= E.dNum
GROUP BY        dNum, sex;

SELECT *
FROM DepEmpStatus;
```

Given DepEmpStatus, but not Department or Employee, we can find general status about employees from each department without having access to confidential information (e.g., address or salary) of employees.

Page 157

Benefits of Views
Simplification: Views can hide the complexity of underlying tables to the end-users.
Security: Views can hide columns containing sensitive data from certain groups of users.
Computed columns: Views can create computed columns, which are computed on the fly.
Logical Data Independence: Views provide support for logical data independence, that is users and programs that access the database are immune from changes in the logical structure of the database

Dropping a view does not affect any tuples from the underlying relation.
DROP TABLE command has options to prevent a table from being dropped if views are defined on it:
Ÿ RESTRICT: drops the table, unless there is a view on it
Ÿ CASCADE: drops the table, and recursively drops any view referencing it

SuperKey: Set of attributes that uniquely identify a relation
R[ABC]
Superkey is ABC
A -> B
Superkey: AC
Superkey: ABC, AB
Minimal Key / candidate Key; A minimal superkey

Directory Access Control (DAC)

- Used to grant privileges to users

Based on Granting and Revoking Privileges

Account level: privileges specified for each account, independent of relations in the account

- Create schema, create table, create view; alter/drop tables; modify

Mandatory Access Control (MAC)

- Classify data and users into various security lasses
- Implement security policy

Role-based Access Control (RBAC)

Module 5: Database Security

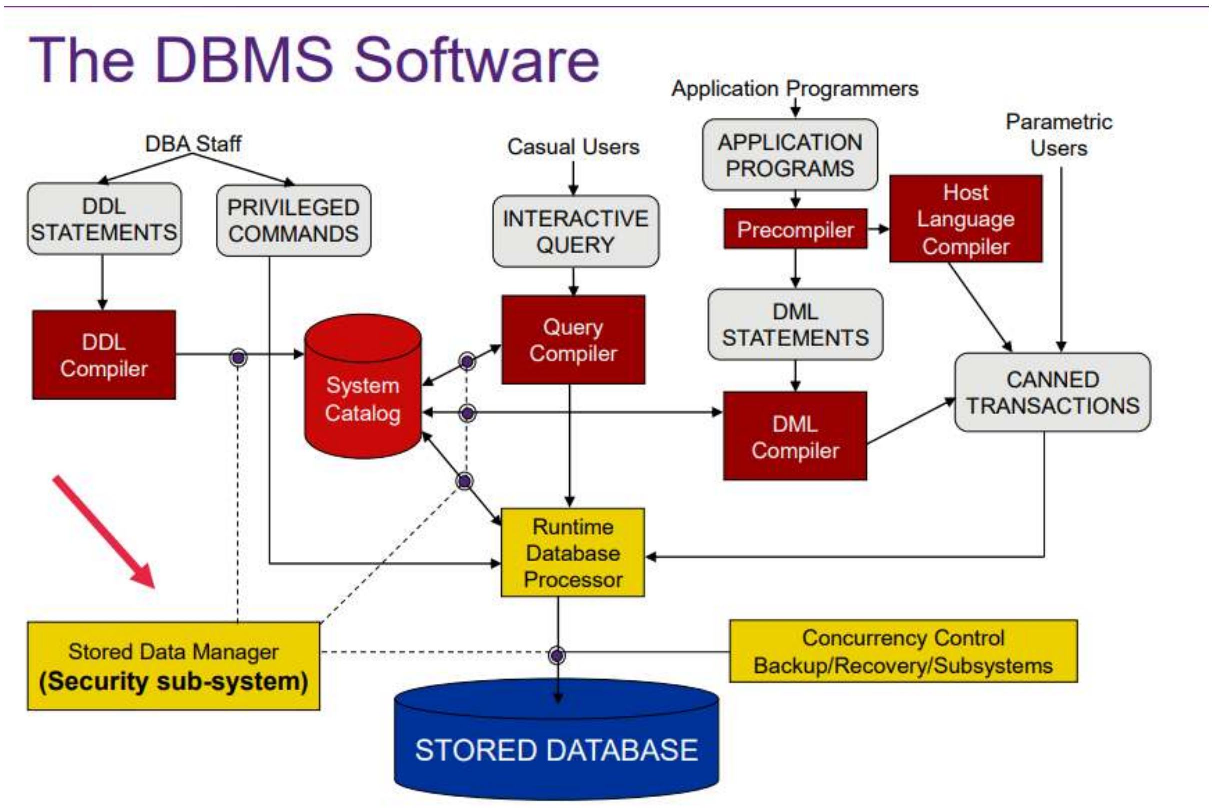
Threats:

- Loss of confidentiality
- Loss of integrity
- Loss of availability

Security - database security concerns how to control the access of data
Privacy - privacy concerns how the data can be used

- It's about the ability of individuals to control the terms under which their sensitive data is acquired and used
    - Preventing storage of sensitive data
    - Ensuring appropriate/authorised use of sensitive data

Security is a required building block for privacy
Privacy is about both directly accessible information almond inferred information



Access Control

- Provide Access only to users who have the right access authority

Inference control

- Must ensure information about individuals cannot be accessed (applies to statistical databases)

Flow Control

- Prevents information from flowing to unauthorised users

Data Encryption

- Used to protect sensitive transmitted data

Access Control Mechanisms

- Authentication: verifying the identity of the person who is accessing your database
- Authorization: determining whether a user should be allowed to execute the translation he's attempting.

Basic Authentication Functions

- User accounts - user must log in using assigned username and password
- Login session - sequence of database operations by a certain user, recorded in system log
- Database audit - reviewing log to examine all accesses and operations applied during a certain time period

Access Control Mechanisms
Direcationary Access Control (DAC)

- Used to grant privileges to users

Mandatory Access Control (MAC

- Classify data ad users into various security classes
- Implement security policy

Role-based Access Control (RBAC)

- Users can be assigned with roles, and then DAC or MAC can be applied

Discretionary Access Control
Based on Granting and Revoking privileges

```
CREATE USER <username> IDENTIFIED BY "<password>";

GRANT <privilege> TO <user>;
```

```
GRANT privilegeName
ON      objectName
TO      {userName |PUBLIC |roleName}
[WITH GRANT OPTION];
```

```
REVOKE privilegeName
ON        objectName
FROM    {userName |PUBLIC |roleName}
```

Two Level of Privileges
Account Level: privileges specified for each account, independent of relations in the account

- Create schema, create table, create view;
- alter/drop tables; modify (insert, delete, update tuples) and select privileges

Relation level: privileges for each relation or view
• Can be specified at attribute level too
• Select (retrieval or read) privilege
• Modification (insert, deleted and update) privilege
• References privilege

```
CREATE SCHEMA AUTHORIZATION oe

   CREATE TABLE Product
      (color VARCHAR2(10) PRIMARY KEY, quantity NUMBER)

   CREATE VIEW RedProduct AS
      SELECT color, quantity FROM Product WHERE color = 'RED'

   GRANT select ON RedProduct TO hr;
```

The above statement

- creates a schema named *oe*
- creates the table *Product;*
- creates the view *RedProduct;*
- grants the SELECT object privilege on *RedProduct* to another user *hr*.

Access Matrix Model

Each relation R is assigned an owner account

Owners can grant (select, modification and references) privileges to other users on any owned relation

	Jake	Aiden	Nelly	Sham
Customer	READ	READ	MODIFY	X
CustOrder	X	X	READ	X
CustOrder.amt	X	READ	READ	X
Supp.status	X	READ	MODIFY	REFERENCE
Supp.address	X	READ	MODIFY	MODIFY

The REFERENCE privilege type grants permission to create a foreign key reference to the specified table.

Each M(i, j) represents the type of privileges (READ, MODIFY, REFERENCE) that a user i has on an object j

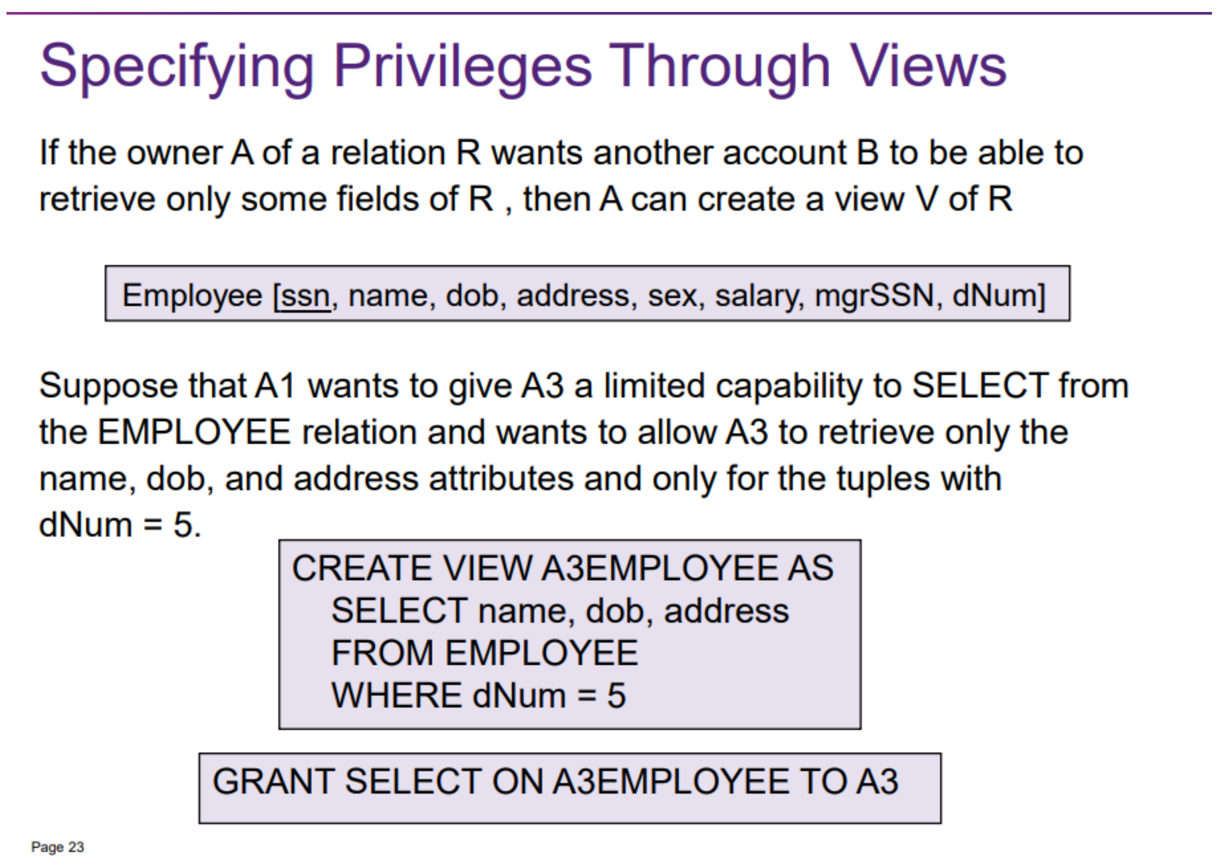
Privilege Revocation/Propagation
Revoking of Privileges

- Useful for granting a privilege temporarily
- Revoke command used to cancel a privilege

Propagation of privileges using the GRANT OPTION
If GRANT OPTION is given to B, B can grant privilege to other accounts without knowledge of owner
DBMS must keep track of how privileges were granted if DBMS allows propagation
Can be very complex

REVOKE [RESTRICT|CASCADE]

If you use CASCADE, it will revoke the privilege and any dependent privileges as a result of your grant
If you use the RESTRICT, and the privilege has been passed, the execution returns an error.



Specifying Privileges Through Views

If the owner A of a relation R wants another account B to be able to retrieve only some fields of R , then A can create a view V of R

```
Employee [ssn, name, dob, address, sex, salary, mgrSSN, dNum]
```

Suppose that A1 wants to give A3 a limited capability to SELECT from the EMPLOYEE relation and wants to allow A3 to retrieve only the name, dob, and address attributes and only for the tuples with dNum = 5.

```
CREATE VIEW A3EMPLOYEE AS
   SELECT name, dob, address
   FROM EMPLOYEE
   WHERE dNum = 5
```

```
GRANT SELECT ON A3EMPLOYEE TO A3
```

Page 23

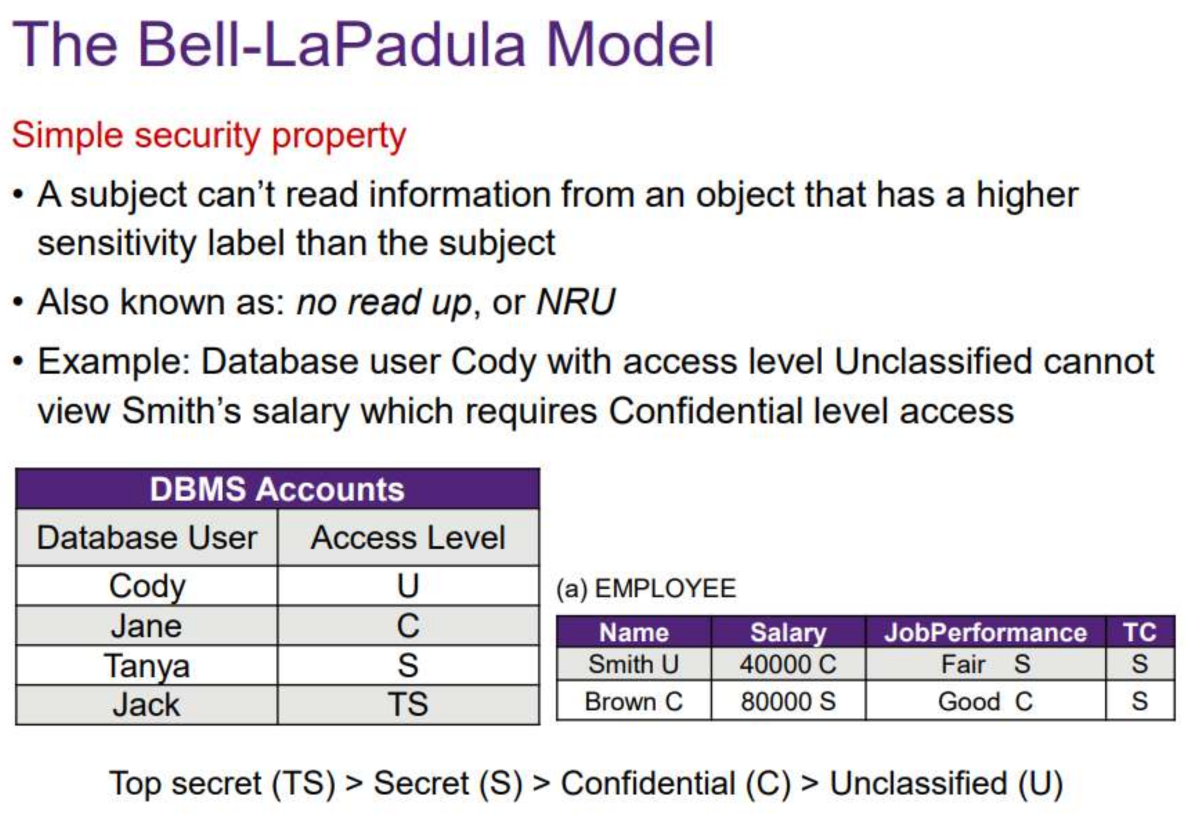
Mandatory Access Control
Addtional security policy that classifies data and users based on security classes

Used for multilevel security

Data and users are associated with security classes

Typical security classes:
- Top Secret (TS)
- Secret (S)
- Confidential (C)
- Unclassified (U)

User: subject
Data: object



The Bell-LaPadula Model

Simple security property

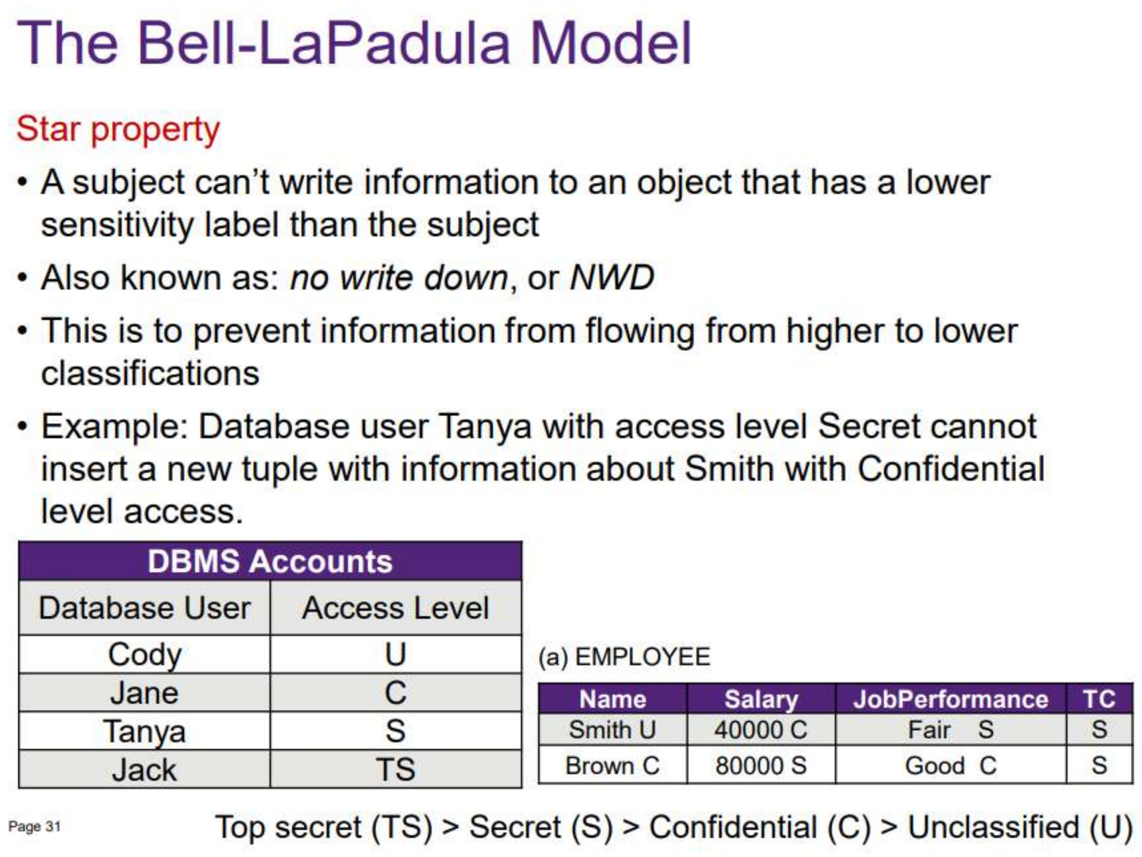
- A subject can't read information from an object that has a higher sensitivity label than the subject
- Also known as: *no read up*, or *NRU*
- Example: Database user Cody with access level Unclassified cannot view Smith's salary which requires Confidential level access

DBMS Accounts	
Database User	Access Level
Cody	U
Jane	C
Tanya	S
Jack	TS

(a) EMPLOYEE

Name	Salary	JobPerformance	TC
Smith U	40000 C	Fair S	S
Brown C	80000 S	Good C	S

Top secret (TS) > Secret (S) > Confidential (C) > Unclassified (U)



The Bell-LaPadula Model

Star property

- A subject can’t write information to an object that has a lower sensitivity label than the subject
- Also known as: *no write down*, or *NWD*
- This is to prevent information from flowing from higher to lower classifications
- Example: Database user Tanya with access level Secret cannot insert a new tuple with information about Smith with Confidential level access.

DBMS Accounts	
Database User	Access Level
Cody	U
Jane	C
Tanya	S
Jack	TS

(a) EMPLOYEE

Name	Salary	JobPerformance	TC
Smith U	40000 C	Fair S	S
Brown C	80000 S	Good C	S

Page 31

Top secret (TS) > Secret (S) > Confidential (C) > Unclassified (U)

Role-based Access Control (RBAC)
Users can be assigned with roles, and then DAC or MAC can be applied

Permissions associated with organisation roles

- Users are assigned to appropriate roles

```
GRANT ROLE full-time TO emp_typ1;
GRANT ROLE intern TO emp_typ2;
```

Roles can be used with traditional DAC and MAC methods

This model has several desirable features, such as flexibility and better support for administration making it an attractive candidate for web-based applications.

SQL Support for Role-Based Access Control

Create/drop roles

```
CREATE ROLE manager;
DROP ROLE manager;
```

Grant/revoke privileges

```
GRANT privilegeName
ON      objectName
TO      {userName | PUBLIC |roleName}
[WITH GRANT OPTION];

REVOKE privilegeName
ON      objectName
FROM    {userName | PUBLIC |roleName}
```

Statistical databases

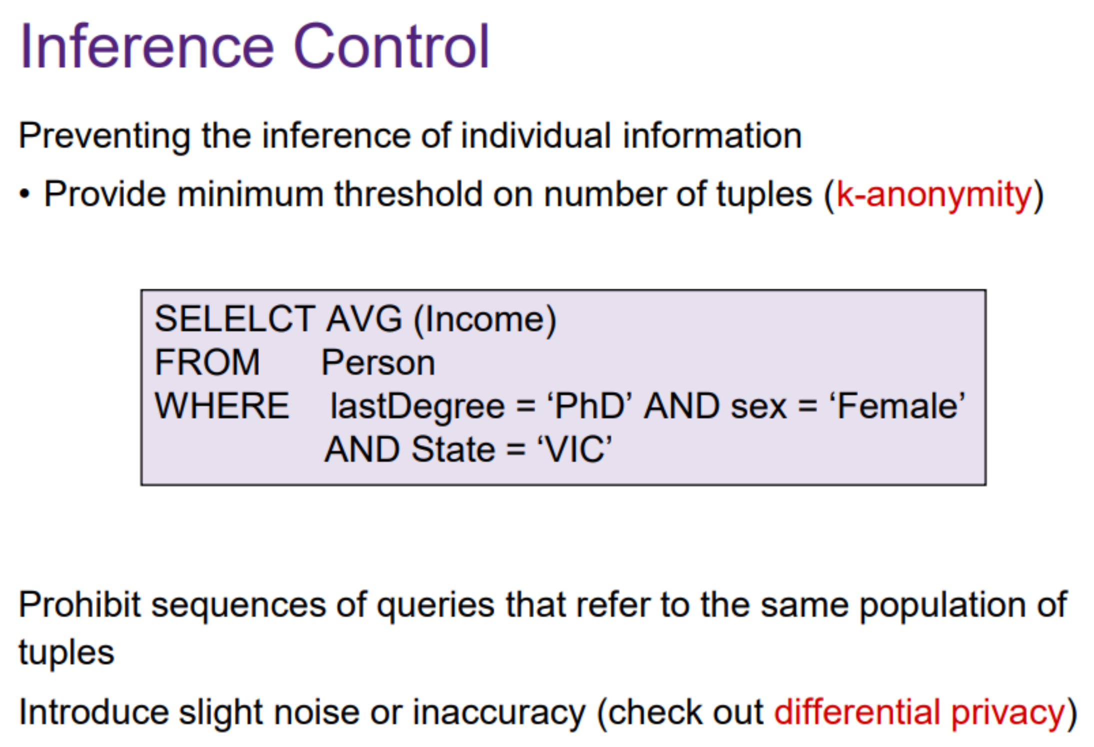
Statistical databases are used to provide statistics about various populations.

Statistical database users such as government statisticians or market research firms are allowed to retrieve statistical information about a population but are not authorized to access detailed confidential information.

Only **statistical queries** that involve statistical aggregate functions such as COUNT , SUM , MIN , MAX, AVERAGE , and STANDARD DEVIATION are allowed

```
Q1:   SELECT COUNT (*) FROM PERSON
      WHERE <condition>;
Q2:   SELECT AVG (Income) FROM PERSON
      WHERE <condition>;
```

Inference Control



Inference Control

Preventing the inference of individual information

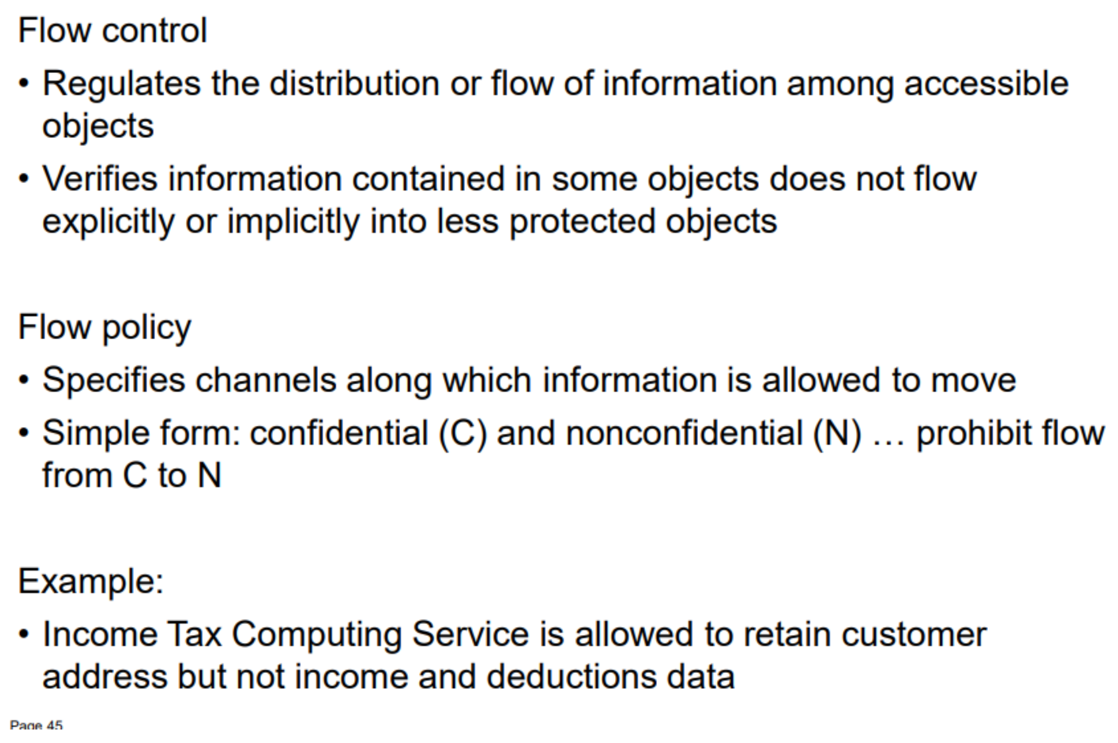
- Provide minimum threshold on number of tuples (k-anonymity)

```
SELELCT AVG (Income)
FROM      Person
WHERE     lastDegree = 'PhD' AND sex = 'Female'
          AND State = 'VIC'
```

Prohibit sequences of queries that refer to the same population of tuples

Introduce slight noise or inaccuracy (check out differential privacy)

Flow Control



Flow control

- Regulates the distribution or flow of information among accessible objects
- Verifies information contained in some objects does not flow explicitly or implicitly into less protected objects

Flow policy

- Specifies channels along which information is allowed to move
- Simple form: confidential (C) and nonconfidential (N) … prohibit flow from C to N

Example:

- Income Tax Computing Service is allowed to retain customer address but not income and deductions data

Data Encryption

Encryption converts data into cyphertext

- Performed by applying an encryption algorithm to data using a prespecified encryption key
- Resulting data must be decrypted using a decryption key to recover original data

Data Encryption Standard (DES)

- Developed by the U.S. Government for use by the general public

Advanced Encryption Standard (AES)

- More difficult to crack

SQL Injection

- One of the most common threats to a database system
- Attackers injects a string input through the application which changes or manipulates SQL statement to attacker's advantage

SQL manipulation

- Changes an SQL command in the application
- Example: adding conditions to the WHERE clause
- Typical manipulation attack occurs during database login

Code injection

- Add additional SQL statements or commands that are then processed

Function call injection

- Database or operating system function call inserted into vulnerable SQL statement to manipulate data or make a privileged system call

Risks Associated with SQL Injection

Database fingerprinting: Attacker determines the type of database being used.

Denial of service: Attacker denies service to valid users.

Bypassing authentication: attacker gains access to database without valid authorization.

Identifying injectable parameters: attacker determines information about backend structure

Executing remote commands: attacker executes harmful commands remotely.

Performing privilege escalation: attacker upgrades their access level

Protection Techniques

SQL injection attacks can occur when SQL statements are used in programs

Bind variables (using parameterized statements)

- Protects against injection attacks
- Improves performance

Filtering input (input validation)

- Remove escape characters from input strings
- Escape characters can be used to inject manipulation attacks

Function security

- Standard and custom functions should be restricted

Normalization – Overview

1NF

- **Outcome**: Removal of non-atomic values from relations.
- **Test**: Relation should have no multivalued attributes or nested relations.

2NF

- **Outcome**: Removal of partial dependencies, which remove some anomalies.
- **Test**: LHS of any non-trivial FD in F^+ is not a proper subset of a candidate key or RHS is a prime attribute

3NF

- **Outcome**: Removal of partial and transitive dependencies, which remove most anomalies
- **Test**: LHS of any non-trivial FD in F^+ is a superkey or RHS is a prime attribute.

BCNF

- **Outcome**: Removal of all anomalies at the cost of not preserving all FDs
- **Test**: LHS of any non-trivial FD in F^+ is a superkey.