ISYS2120

# Week 1

**Data**

* Fact that can be recorded
  + Important for users
  + Requires persistent management

**Database**

* A collection of data
  + Usually large in quantity
  + Normally contains all the information necessary to operate an organization

**Database Management Syustem (DBMS)**

* Software package designed to store and manage one or more databases
  + Allows shared access from many programs

**System Catalogue, Data Dictionary**

* Storing descriptions of the format of the data
* Often also called “metadata”

**Relational DBMS**

* Most DBMS stores table specific information that are linked via special key
* Nominalized database

**Instance**

* An instance is the content of the database at a single time
  + Specific values which describe a specific situation in the world
  + Every update changes the instance

**Schema**

* The schema describes the **Structure** of the data in a database
  + What tables exists
  + What attributes are present and in what form and datatype
* It is also called “integrity constraints” which restricts the possible instances that go against logical structure

“**instance at any given point must fit the pattern of the schema”**

**Data Design**

* Decision of schema is a very important task which focuses on what information is necessary and in what format.
* Decisions of such charasteristics are called “Data design”
* Stages of Data Design
  + Fist produce a conceptual or semantic model
  + Translate them into a relational schema
  + Evaluate the schema for quality and improve it as needed

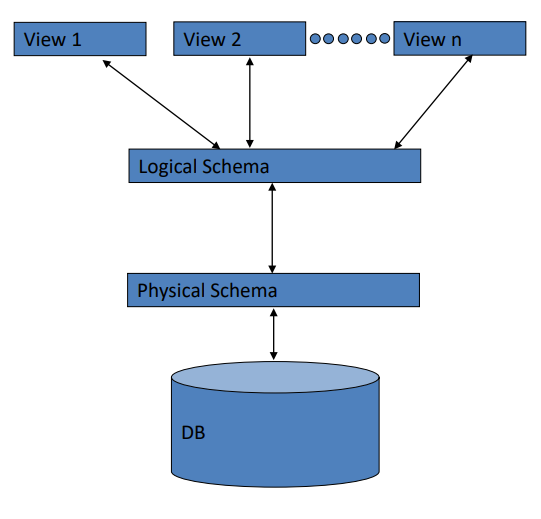
**Data Definition language**

* Used to define the schema
* Allows one to tell DBMS what tables exist and what structure they have
* Create Drop or Alter the Relation Schema

**Data Manipulation Language** (DML)

* Used to access and change data in DBMS
  + Update/Modify
  + Retrieve
  + Insert
  + Delete

**Data Control Language (DCL)**

* Commands that control database, including **administering** **privileges** **and users**

**VIEW:**

* Description of how a user sees the data

**Logical schema**

* Definition of the structure of data as it is shared among all users

**Physical schema**

* Description of the files and indezes used for storage on disk

**Data Independence**

**Logical data independence**

* Protection from changes in logical schema
  + Introducing an extra column in a table

**Physical data independence**

* Protection from changes in physical structure and location of data

Data independence is one of the most important benefit of using a dbms

**Roles within DBMS**

**End Users**

* End users are people who do something that advances the organization’s purpose
  + They run applications that **present** the data to them and allow them to **make controlled changes**
* **Categories**
  + **Offline naïve users**
    - People who gets reports based from the DBMS status
      * Store manager getting weekly profit report
  + **Parametric naïve users**
    - People who execute pre-written applications
      * Deli managers who run application to reorder some items
  + **Ad hoc users** 
    - People who explore the data
      * Division manager looking for trendsA

**Application Programmers**

* IT professionals who produce the application that end users can run
  + This usually fit into a broader software development process, with system analysts, project managers, testers, etc.
* Programmers need to understand how to create an application that accesses data through a DBMS
  + As well, they need a range of software engineering skills
    - Quality assurance
    - SDLC processes
    - User interface

**Database Administrator**

* Responsible for management of Organization’s Databse for effective and efficient use of resources in proving access to data
* Example tasks
  + Design local/physical schemas
    - Make trasde-offs between different choices, to get good performance for all users, at reasonable cost in hardware and software
  + Handle security and authorization
    - Set up accounts and permissions
  + Data availability, crash recovery
    - Make sure backups are taken, and used when needed
  + Database tuning
    - Monitor performance and adjust parameters or redesign schemas as needed

**Files vs DBMS**

|  |  |
| --- | --- |
| **Files** | **DBMS** |
| Data definitions repeats in each program | Data is defined once in the central data dictionary of a BDMS |
| You have to program file readings and line managements manually | Declarative query is easy to read, efficient evaluation due to automatic optimizations |
| Data Integrity can be made from anywhere including incorrect input or missing synchronization | Violation of integrity is only possible in a relational sense:   * Record is deleted from the file with the product without checking the orders first |
| You need to keep track of which programs use which files, and modify all those programs when the file structure changes | You only need to change the description in the database, not in the program that use the data |
| Only a file accessibility is controllable | Declarative access control regarding individual database users and user groups  **Views** |

# Disadvantages of using file system

* Data redundancy and inconsistency
  + Multiple file format, duplication of information in different files
* Difficult in accessing data
  + Need to write algorithm each time you want to access a file
* No central authority, or security
  + Different programmers might want different choices of files and formats, and there is no easy way to enforce organizational control over the valuable data
* Integrity problems
  + Integrity constraints become a part of program code
* Concurrent access by multiple users may lead to undesired outcomes or unsynchronized file managements
* Security problems
* Atomicity of updates
  + Failure may leave database in an inconsistent state with partial updates carried out

SQL statement is in the “What information is needed” than “How to retrieve information”

**SQL Select Statement**

**SELECT** List the columns that should be returned from the query

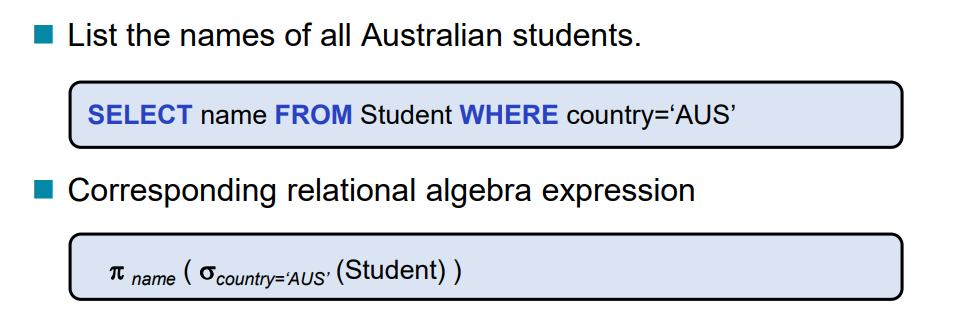
**FROM** Indicate the tables from which data will be obtained from

**WHERE** Indicate the conditions to include a tuple in the result

**GROUP BY** Indicate the categorization of tuples

**HAVING** Indicate the conditions to include a category

**ORDER BY** Sorts the result according to specified criteria



**SELECT**

* **Distinct** To force the elimination of duplicates
* **All** To allow duplicates to be included
* **As** Renaming relations and attributes using the **as** claese
  + - Only in the front end, does not change the name of actual back end attribute name
* **Aggregate Functions**
  + **Avg** Average Value
  + **Min** Minimum value
  + **Max** Maximum value
  + **Sum** Sum of values
  + **Count** Number of values

**FROM**

* **JOIN operations**

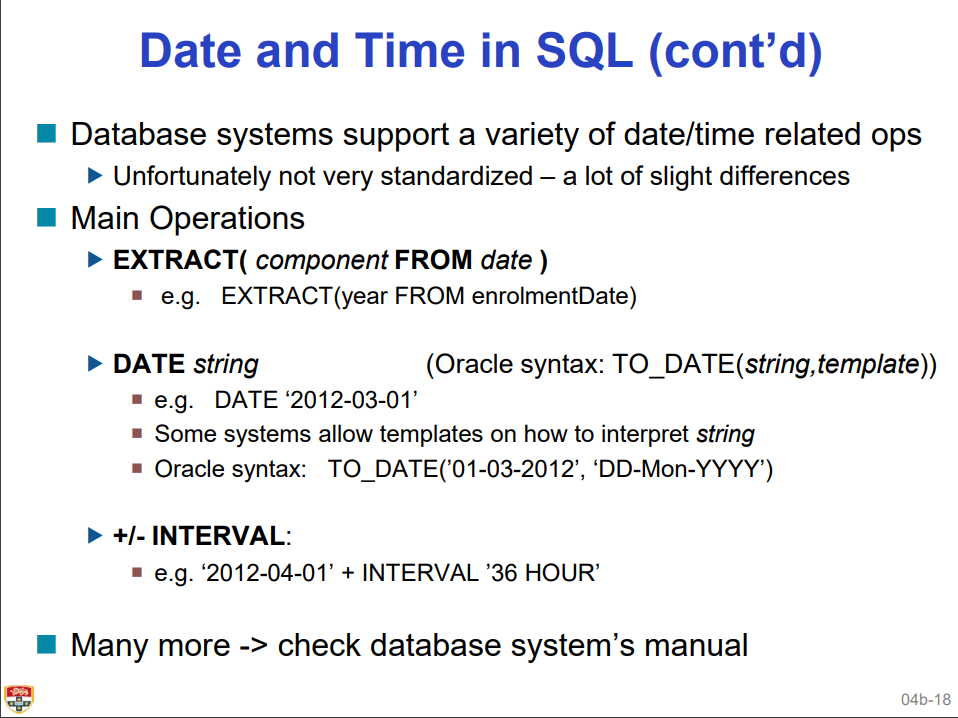
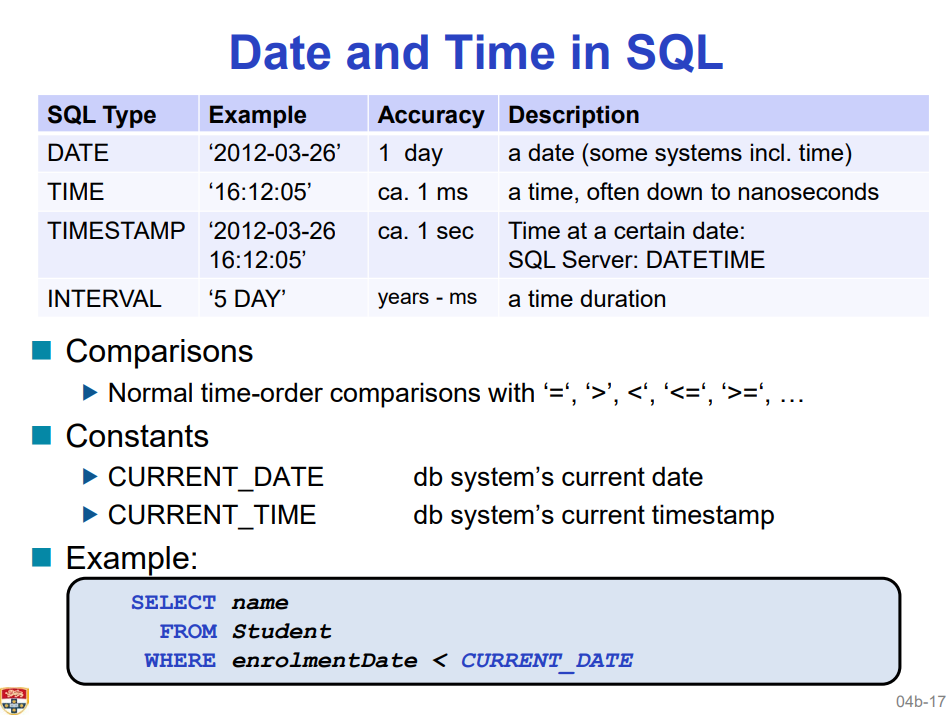
**WHERE**

* Comparison operators in SQL
  + =, >, <, <=, !=, <>
* **AND, OR, NOT**
* **BETWEEN** \* **AND** \* Allows specific range definitions
* \* **LIKE “”** Allows String Wild Card operations
  + % Any number of Substrings
  + \_ One Character
* **||** String Concatenations

**ORDER BY**

* **ASC** Ascending order (default)
* **DESC** Descending order

**Date and Time in SQL**



**Null**

* Aggrivate functions except Count() ifnores null values on the aggregated attributes
* **PRO**
  + Useful because using an ordinary value with special meaning does not always work
    - Put -1 for instance, if you want to put -1 for unknown cell value, then if you want to call aggregate function sum() then it goes to hell.
* **Cons**
  + Null causes complications in the definition of many operations

Database Design Sequence

**Requirement Analysis**

* Understand
  + What data is to be stored
  + What applications must be built
  + What operations are most frequent

**Conceptual Design**

* Develop
  + High level description of the data closely matching how users think of the data
  + Works as communication vehicle

**Logical Design**

* Convert
  + Conceptual design into a logical database schema

**Physical Design**

* Convert
  + Logical schema into a physical schema for a specific DBMS and tune for app

**Conceptual Data Model**

**Aim:** specification of database schema

**Conceptual Data Model**

* Technique for understanding and capturing business information requirements graphically
* It does not imply how data is implemented, created, modified, used or deleted
* Focuses on the internal relationships between databases

**Entity Relationship Model**

* High level graphical representation of what Data needs to be contained in the system
* Associations among different categories of data within a business or information system
  + What are the **entities** and **relationships** in the enterprise

**Entity**

* Object about wghich you want o gather and store data
* Distinguishable from other entities

**Entity Type**

* Collection of entities that share common properties or characteristics
  + Eg: Students, courses , accounts
  + Rectangles represent entity type

**Attribute**

* Describes one aspect of an entity type
  + People have names and addresses
  + Ellipses represent Attributes

**Entity Type**

* Described by a set of attributes
  + Descriptive properties possessed by all members of an entity type

**Domain**

* Possible values of an attribute

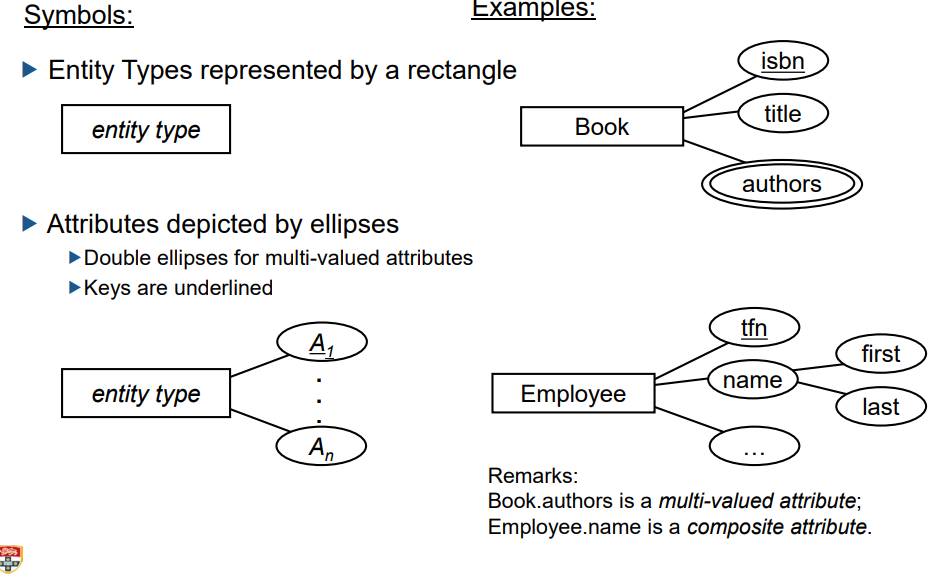
**Key**

* Minimal set of attributes that uniquely identifies an entity in the set]
* Underline in the attribute represents a primary key **PK**

**Entity Schema**

* Entity type name, attributes and PK

**ER Diagram representation**



**Relationships**

* Relates two or more entities
* Number of enttiies is also known as the **degree** of the relationship

**Relationship Type**

* Set of similar relationships
* Shown with a Diamond

**Distinction**

* Relation set of tuples
* Relationship describes relationship between entities

**Relationship Attributes and Roles**

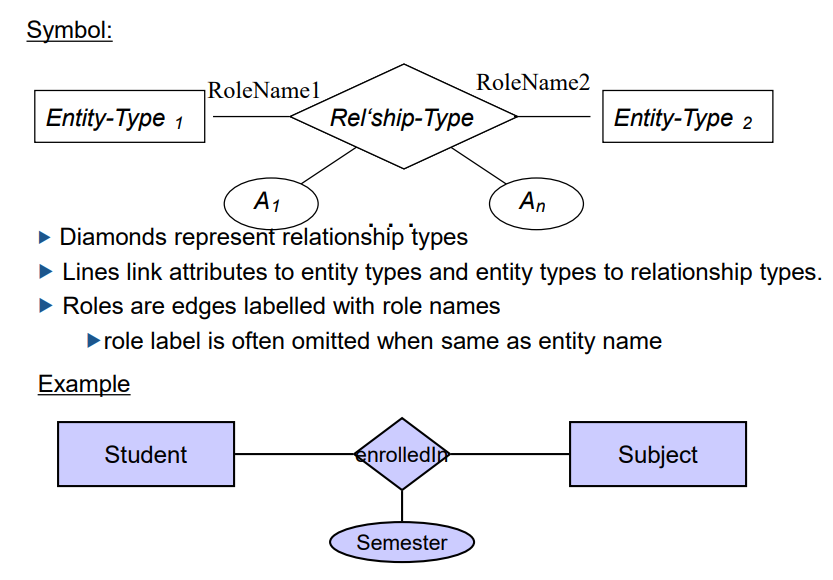
**Relationship-Attribute**

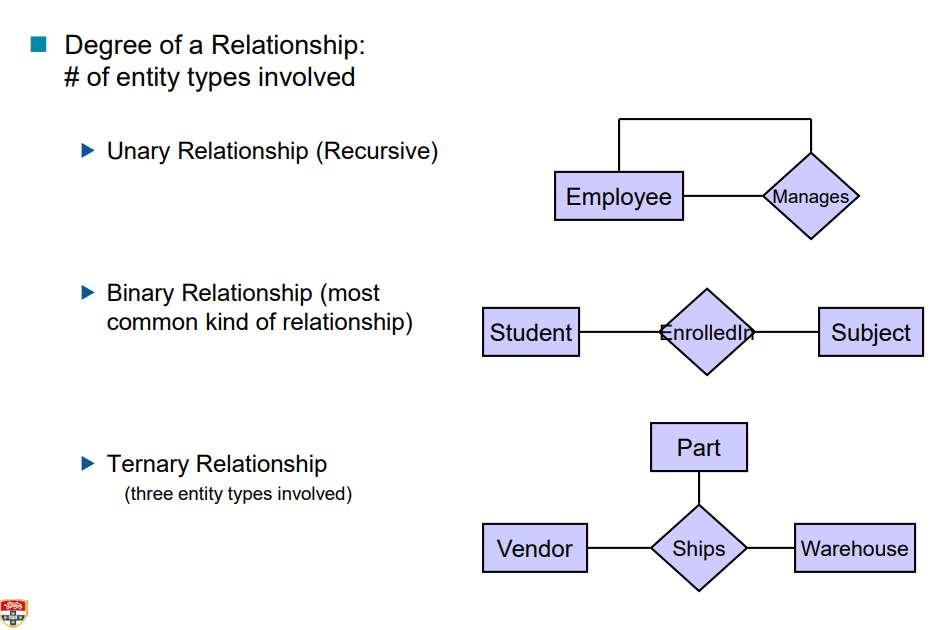
* Relationships can also have additional properties

**Relationship-Role**

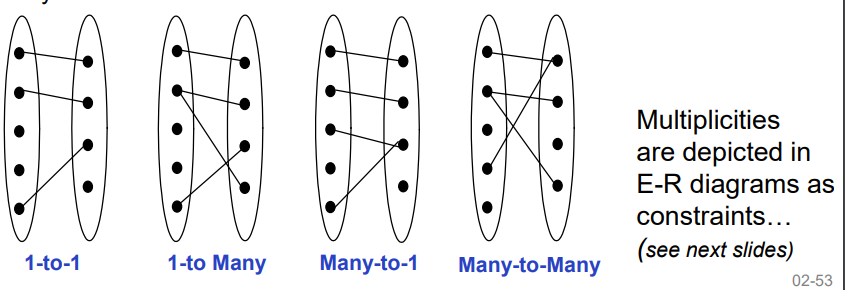
* Each participating entity can be named with an explicit role

**Diamonds** represent **Relationship Type**

****

****

**Relationship Degree is the number of entity types involved**

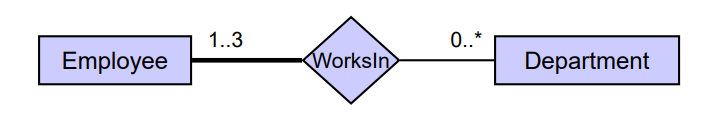
**Multiplicity**

**Model comes from domain knowledge**

* Conceptual data model should indicate whatever we know about the organizations business rules

**Different types of relationships**

* **Key Constraint**
  + **AT MOST ONE**
  + Shown as a thin arrow from key side to the relationship diamond
* **Participation Constraint**
  + **AT LEAST ONE**
    - **Thick line from entity type that must participate to relationship Diamond**
* **Combination of Participation and Key Constraint**
  + **EXACTLY ONE** 
    - **Thick arrow from key entity type that must participate to relationship diamond**
* **Cardinality Constraints**
  + **Specification** of **Maximum and minimum** number of entity that is able to participate.

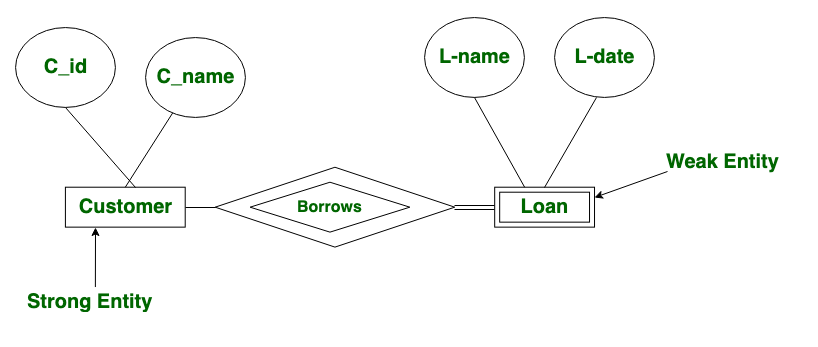
****

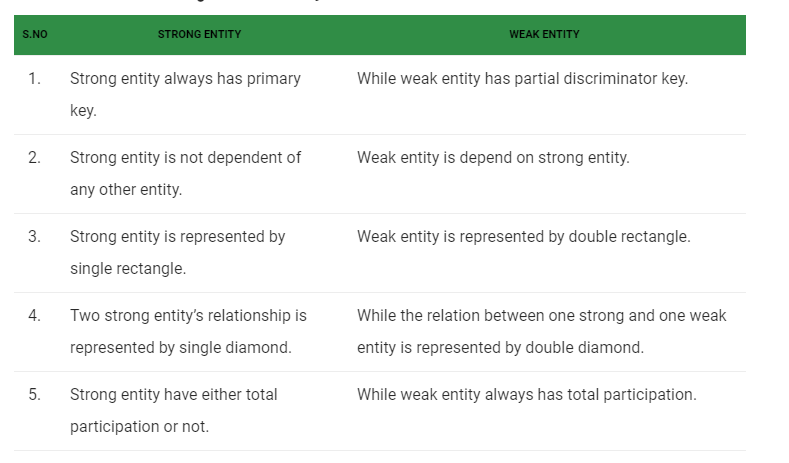
**Employee** works at **1-3 department**

**Department** can have **0 to many** employees

**Weak entity**

* These are an entity that cannot exist without existance of a dependent entity
* These entity have **TOTAL PARTICIPATION CONSTRAINT** in it’s idenfigying relationship ith owner identity





**Descriminator** of a weak entity type is the set of attributes that distinguishes among all the entities of a weak entity type related to the same owning entity

# Week 2

**Data Model**

* A collection of concepts for describing data
  + Structure of the data
  + Operations on the data
  + Constraints on the data

**Relation**

* Two dimentional tables of data
  + Row/ record
  + Column / attribute / field

**Requirement to becoming a relation (1NF)**

* Every relation must have a **unique name**
* Attributes in a table must have **unique name**
* All tuples in a relation have the **same structure**
* Every attribute value is **atomic**
* Every row is **unique**
* The order of the rows is **immaterial**

**Relation Schema**

* Specifies name of the relation as well as it’s attributes and data types

**Relation instance**

* Set of tuples for a schema, Table

If you think about relation as a table **object**, then relation schema and relation instance is completely Different

#rows = **cardinality**

# Columns = **degree (arity)\_of relation**

**Relational Database**

**Data Structure**

* Set of Relation instances

**Creating Relations in SQL**

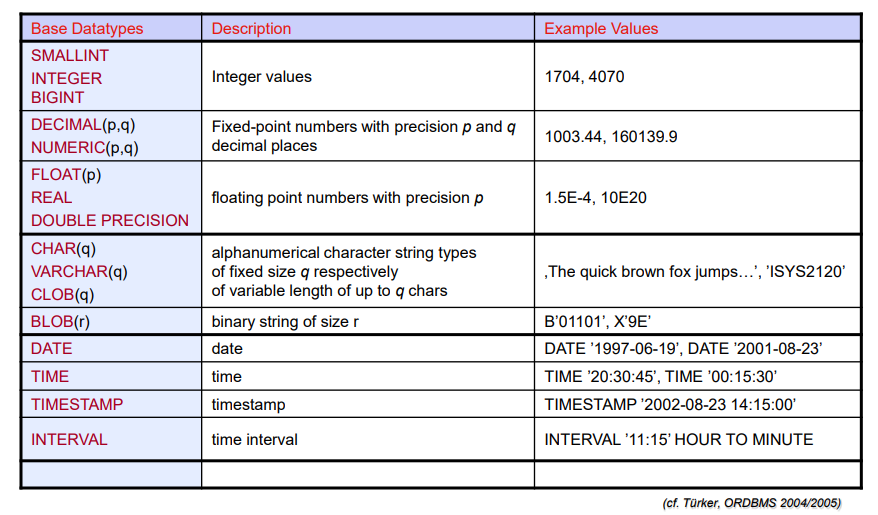
**CREATE TABLE** table\_name (

Attribute\_name data\_type **Integrity\_constraints**

)

**Deleting Tables**

**DROP TABLE** table\_name



Modifying Instances using SQL

**INSERT INTO** table [“(List of Columns) “] VALUES (“List of expressions”)

INSERT INTO Student (sid, name) VALUES (480222279, ‘Josh’)

**UPDATE** table\_name SET column = ‘expression’ ,{ column = ‘expression’ } [WHERE condition]

UPDATE Student SET address = “55 Duke Street’ WHERE sid = ‘490223789’

**DELETE FROM** table [**WHERE** condition]

DELETE FROM student WHERE name = ‘smith’

**Integrity Constraints (domain constraints)**

* Conditions that must be true for any instances of the database of that attribute
* Legal instance of a relation is one that satisfies all specified IC

**INTEGRITY CONSTRAINTS**

**NON-NULL**

* No value in a given column can be null

**PRIMARY KEY**

* Unique, minimal identifiers in a relation
* There may be several **candidate key** to choose from
* DOES NOT ALLOW NULL
* **PRIMARY KEY (sid,cid)**

**UNIQUE**

* A attribute(s) that can only have a unique values/pairs/combinations
* ALLOWS NULL
* **UNIQUE (sid,grade)**
  + A single student can have a certain grade **only once**

**Foreign keys**

* **Reference Integrity**
  + Must refer to a **existing candidate key of the parent relation**
* Identifiers that enable a **dependent relation** to refer to its **parent relation**’
  + **FOREIGN KEY** (sid) **REFERENCES** Student
* ALLOWS NULL WITH CERTAIN CONDITIONS

**CASCADE**

Update values according to any changes made in the parent tuple

**REFERENCES** Professor

**ON UPDATE CASCADE**

**ON DELETE SET DEFAULT**

**ER DIAGRAM**

**Relations (table)** correspond to entity types

**Rows** correspond to Entity’s **cardinality**

**Column** corresponds to **attributes**

**Entity Types**

* **Simple Attributes**
  + Directly mapped onto the relation
* **Composite attributes**
  + Flattened out by creating a separate field for each component attribute
* **Multi-valued attribute**
  + Becomes a separate relation with a foreign key taken from the superior entity
  + Becomes a weak entity

**Mapping of Relationship Types**

* **Many to Many**
  + Create a new relation with primary key of the two entity types as it’s primary key
* **One to many**
  + Primary key on the one side becomes a foreign key on the **many side**
* **One to one** 
  + Primary key on the mandatory side becomes a foreign key on the optional side
* **Relationship attributes**
  + Becomes fields of either the dependent, respectively new relation

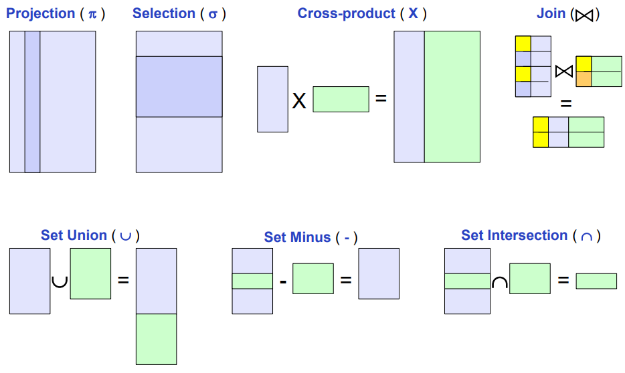
# WEEK 3

Relational Algebra

* Defines some basic oeprators that can be used to show general expression of a query

The aim of Relational Algebra is to allow simpler specification of Query without worry about implementation

Further extraction of “what” than “how”

Relational Algebra

**Union**  tuples in relation 1 or in relation 2

**Intersection** tuples in relation 1 as well as relation 2

- **Difference** tuples in relation 1 but not in relation 2

**Projection** Deletes unwanted columns from relations

**Selection** Selects a subset of rows from relation

X **Cross** P**roduct** allows us to fully combine two relations

**Join** To combine matching tuples from two relations

**rename** rename fields or even whole relation



Get the names from student relation where country = Australia

Conditional join

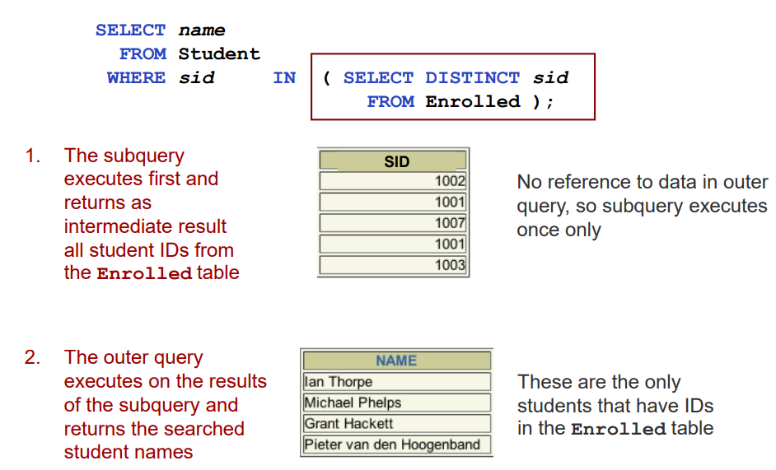
Conditions can be any comparative operations (=,>,<,>=,!=)

Conditions are called **EQUI-JOIN** if conditions are all equalities

Conditions by definitions are called **THETA-JOIN**

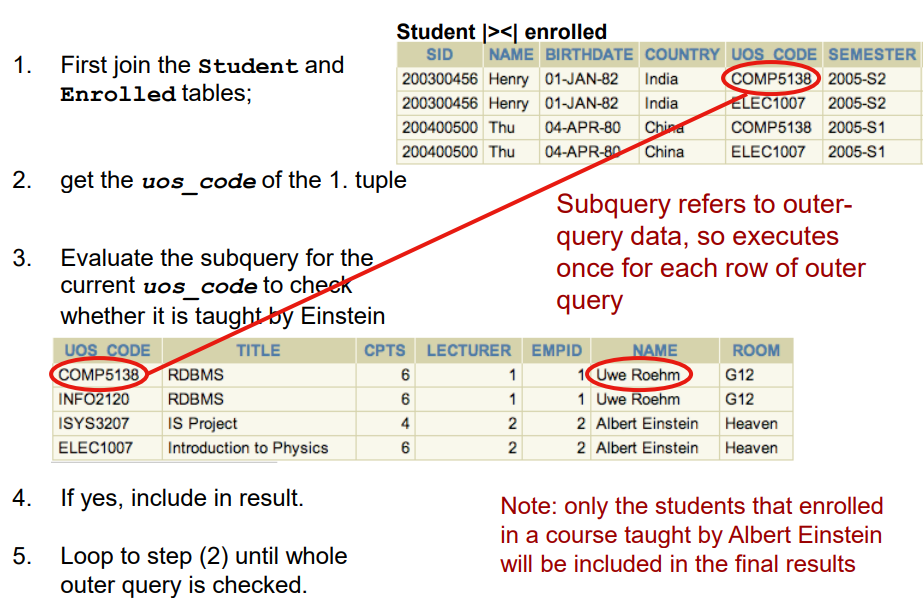


Get all the Entity from Enrolled who were enrolled in ISYS2120, show sid as Student in the display

**Subqueries**

**Correlated Subquery**

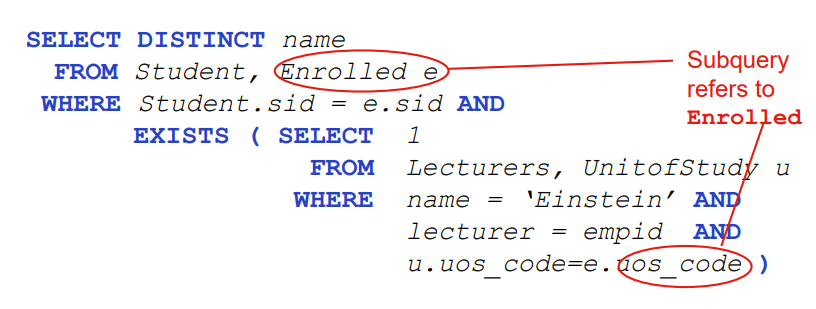
* Do not depend on data from the outer query
* Execute once for the entire outer query

**Correlated Subquery**

* Make use of data from the outer query
* Execute once for each row of the outer query
* Can use the EXISTS operator

**Find all the students who have enrolled in the lecture**

**Given by ‘Einstein’**

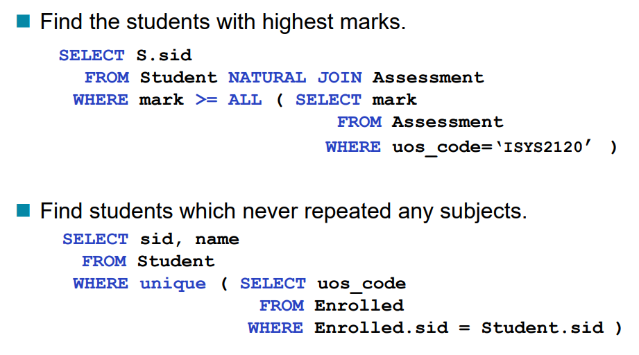
****

**IN**

Compares a value **v** with a set returned from a subquery, if **v** is one of the element in subquery, it returns true

**EXISTS**

Checks whether there is a returning value from a **correlated subquery**

**SET comparison operators**

**Not exists**

* Tests whether the returning set is empty

**Unique**

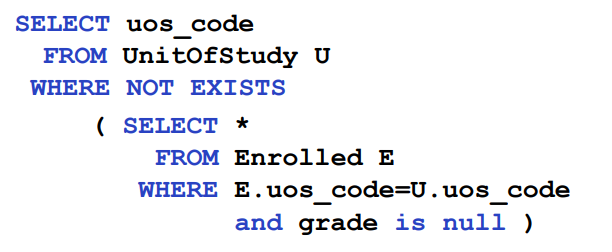
* Tests whether a subquery has any duplicate tuples in it’s result

**All**

* Tests whether a predicate is true for the whole set

**Some**

* Tests whether some comparison holds for at least one set element

****