FIT2094 - Databases - Lecture note 1-12

Databases (Monash University)

FIT2094 - Databases

Week 1:

* to complete*

Week 2:

Relational Model

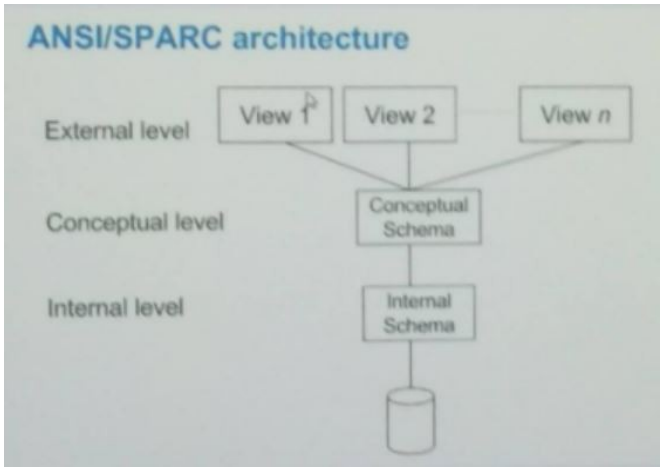
- Relation
  - Tables
  - Has a header (schema) and body (instance)
- Domains
  - Defines a property of an attribute
- Tuple
  - Row in a relation
- Attributes
  - Columns of the relation
  - For every attribute there is a domain
- Degree
  - Number of attributes in a relation
- Cardinality
  - Number of tuples in a relation
- Concepts of Relational model
  - Relation: Abstract object
  - Table: Pictorial representation
  - Storage Structure: “real thing”
- Two parts
  - Head (Relational Schema)
    - Each attribute corresponds to underlying domain
    - R = Relation
    - Ai = Attribute
  - Body (Relational Instance/State)
    - Consists of time varying set of n-tuples
    - M = number of tuples = relation cardinality
    - Each n-tuple is an ordered list of n values
    - -n = number of values in tuple = relational degree
- Relational Properties
  - No duplicate tuples
  - Tuples are unordered within a relation
  - No ordering of attributes within a tuple
  - Tuple values are atomic - can not be divided
- Functional Dependencies
  - Set of attributes X functionally determines an attribute Y if for each X value, there is exactly one Y value in the relation
  - Denoted: X → Y

- Data Anomalies
    - Update
    - Delete
    - Insert
- Keys
    - Superkey: Single attribute or set of attributes, X that uniquely identify a tuple in a relation R
        - Multiple keys
        - I.e. X → {R}
    - Candidate Key: A minimal set of attributes X that uniquely identifies each relation R
        - Subset of superkey
        - any column or a combination of columns that can qualify as unique key in database
        - There can be more than one in a relation
        - I.e. X → {R} AND for every subset Y or X, Y must not be a superkey
        - Y → {R} must NOT hold
    - Primary Key
        - Chosen candidate key
        - Also a composite key (made up of multiple attributes)
        - Can not be a null value
        - Only one
    - Foreign Key
        - An attribute/s in a table that exist in the same, or another table as a Primary key
        - Link between two relations
        - Values of a foreign key must match a primary key or be null
    - Surrogate Key
        - An added attribute for use as a primary key
    - Composite Key
- Algebra
    - SELECT
        - Denoted: σ(sigma)
        - Collect data in entire tuple
        - It's a selection if the columns are not specified, but if they are specified then it can be projection
        - If not all columns are selected , it is a projection
        - SELECT[expr](rel)
    - PROJECTION
        - Denoted:π(pi)
        - Collect data in attributes
        - Can be multiple columns
        - PROJECT[A,B,C](Rel)
    - RELATION = |x|
    - JOIN
        - Table1 x Table2
        - Rel1 Join[expr] Rel2

- NATURAL JOIN
  - Join tables
  - Delete tuples where keys do not match
  - Delete duplicate columns
- OUTER JOIN
  - Ignore rows with incomplete information (eg. student in student table having no associated marks in the mark table)
- FULL OUTER JOIN
  - Instead of omitting incomplete rows, put null as the unknown cells
- LEFT OUTER JOIN
  - Ignore incomplete tuples from the right table
- RIGHT OUTER JOIN
  - Ignore incomplete tuples from the left table
- DIFFERENCE
- PRODUCT
- INTERSECTION

-

Week 3:



Database life cycle:

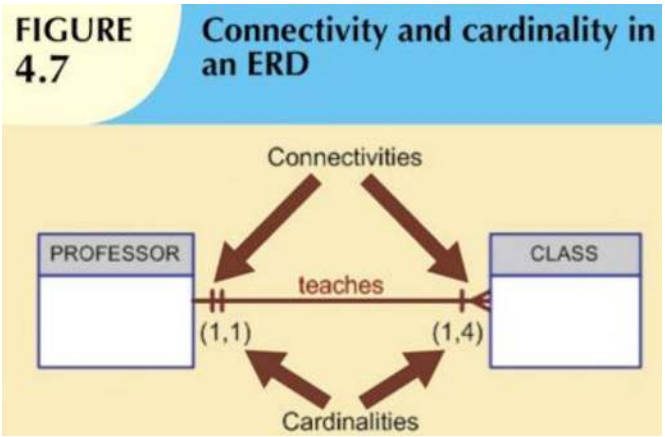
- Requirements Definition → Conceptual Design → Logical Design → Physical Design

Requirement Definition:

- Identify and analyse user views
- User view is report
- Corresponds to external level of architecture
- Output is statement of specifications

Entity Relationship Model

1. Basic components
    a. Entity
    b. Attribute
    c. Relationship



Entity Relationship Diagram:

- Chen's Notation
    - Semantically rich
    - Complex diagram
    - Pure conceptual level
- Crow's food
    - Less semantics
    - Simple diagram
    - Conceptual and logical levels

Types of Attributes

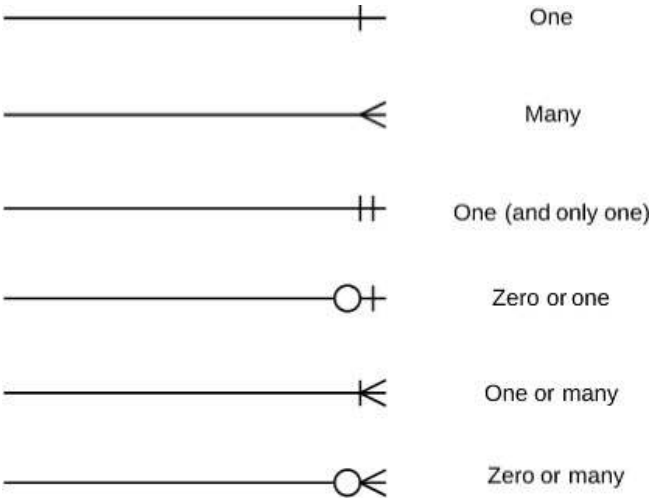
- Simple:
    - Cannot be subdivided
- Composite:
    - Cannot be subdivided into additional attributes
- Single-valued
    - Can only have single value
- Multi-valued:
    - Can have multiple values

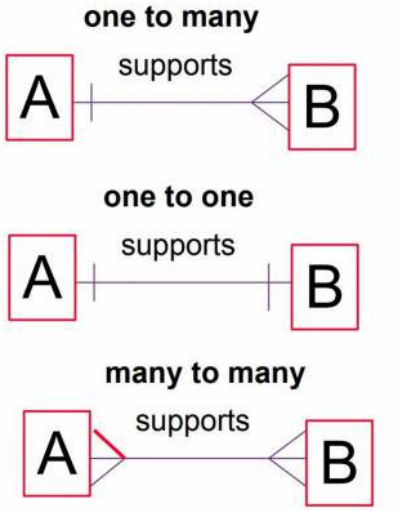
Cartesian Product - Multiply rows
Equal Join - Primary Key joined to another table with the equivalent of a foreign key
Natural Join - same as equal, remove repeating rows

When a primary key of a parent table, is a part of the primary key of the child table (child dependent on parent), there is a strong relation (identifying relation)

To resolve a many:many scenario, a composite entity would be introduced (to break them)

Recursive relationship: a relationship is between 2 entities, in a recursive relationship the same entity participates in the relationship (eg, an





employee who is also a manager)

Week 4:

What not to put in conceptual models?

- Primary Key
- Foreign Key
- Surrogate Keys

Why is this?

- Because the purpose of a conceptual model is that the model is database independent

Process for finding identifiers (keys):

Identify superkeys → Set of minimal superkeys → Candidate keys → Choose primary key

Properties of Relations

- Unique name in the database
- Each row/tuple is unique
- Each column has a meaningful name
- Order of attributes is immaterial
- Order of tuples is immaterial
- Entries are atomic (single-valued)

Transforming ER into relations

1. Map strong (regular) entities
2. Map weak entities
3. Map binary relationships
4. Map associative entities
5. Map unary relationships
6. Map ternary relationships
7. Map supertype/subtype relationships (not part of the unit)

Map Regular Entities:

- Composite Attributes

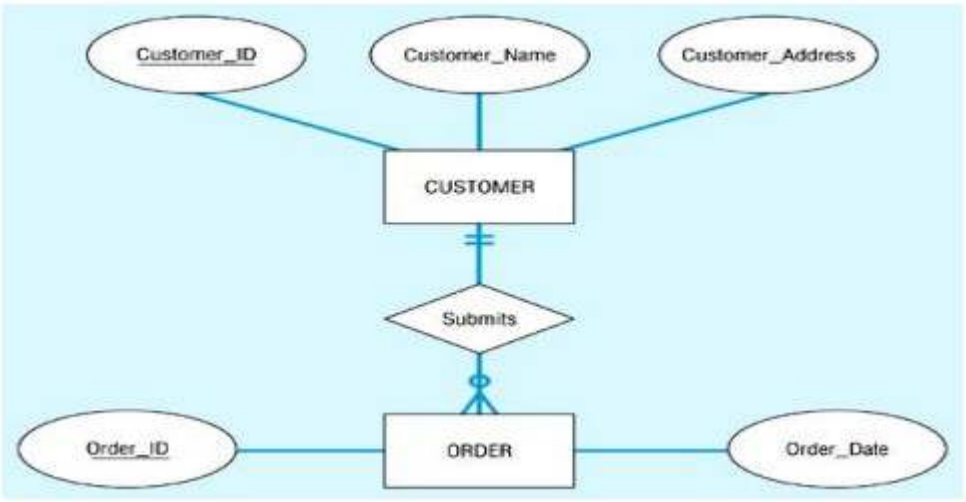
    - Only the simple component attributes of the composite attribute are included in the new relation
    - Compared to composite attributes, simple attributes improve data accessibility and help maintain data quality
    - MAPPING:
        - No asterisk means can not have null values
- Multivalued Attributes
    - Two new relations are created
        - First relation contains all the attributes of the entity type except the multivalued attribute itself
        - Second relation contains two attributes that form the PK (One of the attributes is the PK from the first relation, which becomes the FK in the second relation and the other is the multivalued attribute
    - There can also be non key attributes in the second relation (depends on data requirements)
    - MAPPING:
        - Take attribute out of entity and make new entity
        - Make a one to many association

Map Weak Entities:

- For each weak entity
    - Create a new relation and include all of the simple attributes as attributes of this relation
    - The PK of the identifying relation is also included as the FK in this new relation
- MAPPING:
    - Take attribute out of entity and make new entity
    - Make a one to many association

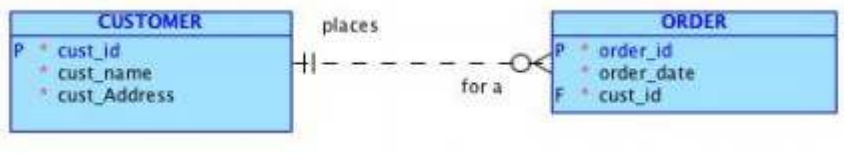
Map Binary Relationships

- Conceptual (1:M)

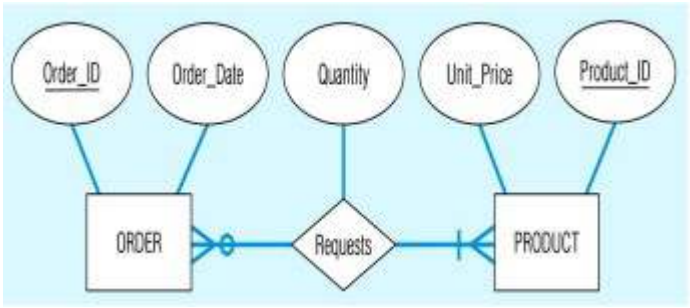


- Logical (1:M)

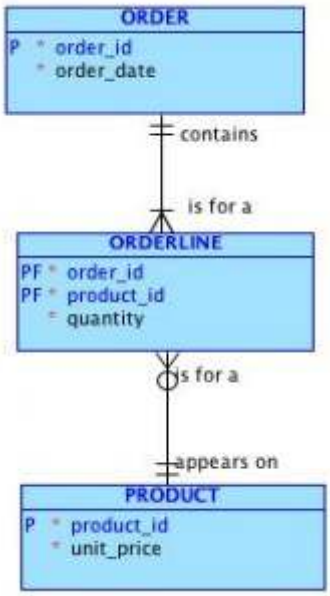
- First create a relation for each of the two entity types participating in the relationship
- Include the PK attribute of the entity on the one-side of the relationship as the FK on the many-side of the relationship



- Conceptual (M:N)



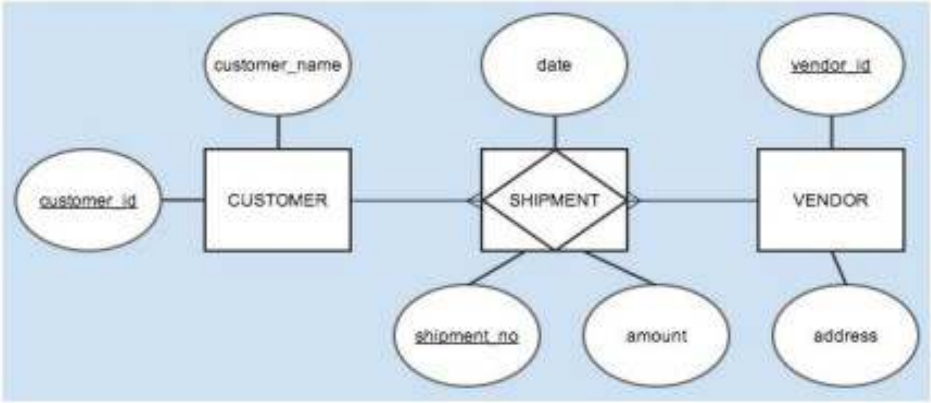
- Logical (M:N)



- First create a relation for each of the 2 entity types
- Then create a new relation and include as FK attributes, the PK attribute for each of the 2 participating entity types - these attributes become the PK of the new relation

- Any nonkey attributes associated with M:N relationship are also included in the new relation

- Map associative entities with an identifier



If key is self-standing it's a strong entity.
If key is taken from parent then its a weak entity.

When both entities are strong the relationship is non-identifying

Week 6: Creating & Populating a Database - DDL

- Data Definition Language - DCL
- Data Control Language - DML
- Data Definition Language - DDL

Common ORACLE data types

Text:

CHAR(size)

- More simpler to use
- Only use when width is fixed

VARCHAR2(size)

- Save less space
- More difficult for system
- Mainly use

Numbers:

NUMERIC

NUMBER(precision, scale)

- Precision: total number of digits
- Scale: how many to the right of the dp
- Mainly use

Date:

DATE

- Julian date and time
- Mainly use

TIMESTAMP

- fraction of a second
- With time zone

ALTER - alters existing object

Week 8: Update, Delete and Transaction Management

UPDATE

- Modifies data in a table

Syntax

UPDATE	Tablename
SET	Columnname = expression [,columnname = expression]
[WHERE	Conditionlist];

- Without the WHERE command, the update applies to all rows in the table

COMMIT and ROLLBACK

- Commit stores changes permanently in the database
- Before a commit has been issued, rollback can be used to restore the database to its condition at the time of the last commit

(only reverts changes from insert and update commands, will not remove tables)

DELETE

- Deletes a row in a table

Syntax

DELETE FROM	Tablename
[WHERE	Conditionlist];

- Again, without where command, all rows in the table will be deleted

ALTER TABLE

- 3 options: ADD a column, MODIFY to change column characteristics, DROP to delete a column from the table

Syntax ADD MODIFY
ALTER TABLE tablename
{ADD | MODIFY} (columnname datatype [{ADD | MODIFY } columnname datatype]);

Syntax to add constraints
ALTER TABLE tablename
ADD constraint [ADD constraint];

Syntax to remove comumn or restraint
ALTER TABLE tablename
DROP {PRIMARY KEY | COLUMN columnname | CONSTRAINT constraintname};

UPDATE

example use:

UPDATE	PRODUCT
SET	P_SALECODE = '2'

WHERE	P_CODE = '1546-QQ2';

Multiple values at once:

UPDATE	PRODUCT
SET	P_SALECODE = '1'
WHERE	P_CODE IN ('2232-QWE', 2341-WDR');

ADDING KEYS

ALTER TABLE tablename
ADD PRIMARY KEY (attribute)

ALTER TABLE tablename
ADD FOREIGN KEY (attribute) REFERENCES tablename2

Transactions Properties (ACID test)

Atomicity:

- All operations of a transaction must be entirely completed or entirely aborted

Consistency:

- It must take the database from one consistent state to another

Isolation:

- It must not interfere with the other concurrent transactions
- Data used during execution of a transaction cannot be used by a second transaction until the first one is completed

Durability:

- Once completed the changes the transaction made to the data must be durable, even in the event of system failure

Transaction Management:

- The management of multiple transactions submitted concurrently by variou users to the same database system.
- Follows the ACID properties
- Transaction boundaries:
    - Start
        - First SQL statement is executed (eg. Oracle)
        - Some systems have BEGIN WORK type command
    - End
        - COMMIT or ROLLBACK
- Concurrency Management

Two main database operations that can be included in a transaction:

- **Read(X)** returns the value stored in data item X
- **Write(X)** changes the stored value of X.

Serial vs Interleaved Transactions:

Serial:

- All operations in one transaction happen first, then the operations of the second transaction
- Will not violate isolation property

Interleaved:

- Occur simultaneously

Problems:

- Lost Update: The value written by T0 is lost. (Go over this one)
- Temporary Update (Dirty Read): T1 reads the dirty value of X. In interleaved, T1 reads a value from T0 that was aborted.
- Incorrect Summary Problem: T1 reads X after subtraction but Y before addition. In interleaved, T1 summarises the result of the first transaction before it's completed.

Management:

- Only read, write, commit and abort are considered
- Can be written as:
  S1: r0(X); w0(X); c0; r1(Y); w1(Y); r1(X); w1(X); c1;
- If transactions cannot be interleaved, there are two possible correct schedules, one that has all operations in T0 before all operations in T1, and a second schedule that has all operations in T1 before all operations in T0.

"A given interleaved execution of some set of transactions is said to be ***serializable*** if and only if it produces the same result as some serial execution of those same transactions".

T0	T1
Read(X) X=X+1 Write(X)	 Read(Y) Y=Y*2 Write(Y) Read(X) X=X+2 Write(X)

Serial

T0	T1
Read(X) X=X+1 Write(X)	 Read(Y) Y=Y*2 Write(Y) Read(X) X=X+2 Write(X)

Interleaved

This is serializable

T0	T1
Read(X) X=X+1 Write(X)	 Read(Y) Y=Y*2 Write(Y) Read(X) X=X+2 Write(X)

Interleaved

This is NOT serializable

- Serializability Theory: an important theory of concurrency control which attempts to determine which schedules are “correct” and which are not and to develop techniques that allow only correct schedules.

Testing for Serializability:

1. For each transaction Ti in the schedule, draw a node labelled Ti.
2. For each conflicted operation, where Tj occurs after Ti, draw an edge Ti -> Tj.
3. The schedule is serializable, if there are no cycles (acyclic).

When there is no cycle, an equivalent serial schedule by:

- Identifying each edge in the graph
- Ordering the schedule so that for an edge Ti -> Tj, Ti occurs before Tj
- If there are no edges, then the operations can be ordered in (almost) any order

Locking:

CONTINUE NOTES

- lock is released when all operations are complete for transactions
- Deadlock occurs when two transactions wait on each other to complete

How to determine if schedules will cause a deadlock?

- Choose a transaction to abort (victim selection)
- Asked to rollback

Week 9: SQL Intermediate

Additional SELECT Query Keywords:

Aggregate Functions:

SOME BASIC SQL AGGREGATE FUNCTIONS	
FUNCTION	**OUTPUT**
COUNT	The number of rows containing non-null values
MIN	The minimum attribute value encountered in a given column
MAX	The maximum attribute value encountered in a given column
SUM	The sum of all values for a given column
AVG	The arithmetic mean (average) for a specified column

- COUNT
  - Can be used in conjunction with the DISTINCT clause
  - Uses one parameter within parentheses, generally a column name such as COUNT(V_CODE) or COUNT(P_CODE)
- MAX and MIN
  - MAX(column name) can be used only in the column list of a SELECT statement
  - Need a nested query, because you need to compute the maximum price first, then compare it to each price returned by the query:
    - The inner query is executed first, outer is executed last
    - SELECT P_CODE, P_DESCRIPT, P_PRICE

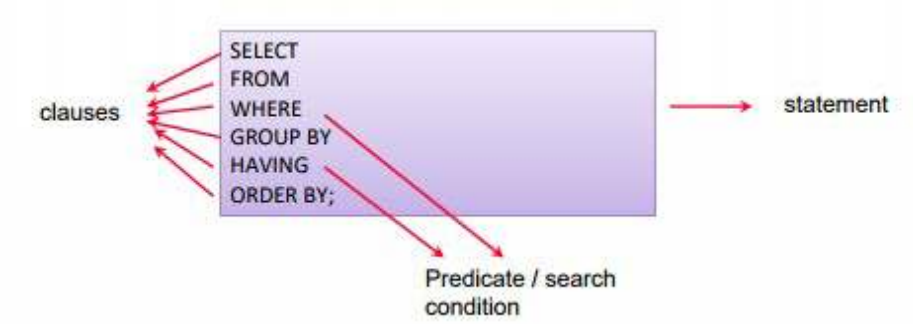
FROM PRODUCT

WHERE P_PRICE = (SELECT MAX(P_PRICE) FROM PRODUCT);

- SUM
  - Computes the total sum for any specified attribute
  - SELECT SUM(CUS_BALANCE) AS TOTBALANCE

FROM CUSTOMER;

- AVG
  - Function format is similar to MIN and MAX, and is subject to the same operating restrictions



Grouping Data:

- Rows can be grouped into smaller collections quickly and easily using the GROUP BY clause within the SELECT statement
- Aggregate functions will then summarise the data within each smaller section

- Syntax:

```
SELECT columnlist
FROM tablelist
[WHERE conditionlist ]
[GROUP BY columnlist ]
[HAVING conditionlist ]
[ORDER BY columnlist [ASC | DESC] ];
```

- GROUP BY clause is only valid when used in conjunction with one of the SQL aggregate functions
- If a GROUP BY clause is used with an aggregate function, the DBMS will apply the aggregate function to the different groups defined in the clause rather than all rows.
- HAVING clause:
    - It is used to put a condition or conditions on the groups defined by GROUP BY clause
    - Much like the WHERE clause for SELECT
    - Applied to the output of a GROUP BY operation
    - Syntax:

```
SELECT unit_code, count(*)
FROM enrolment
WHERE mark IS NULL
GROUP BY unit_code
HAVING count(*) >1;
```

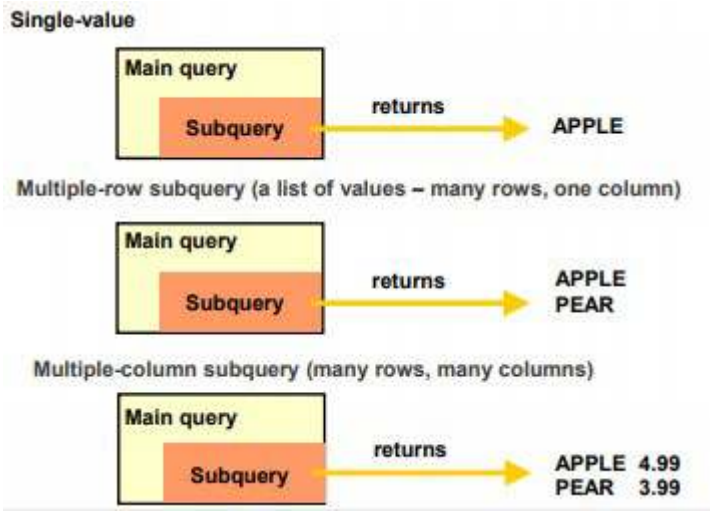
- The logic of the process:
    - All rows where mark is NULL are retrieved (due to the WHERE clause)
    - The retrieved rows then are grouped into different unit_code
    - If the number of rows in a group is greater than 1, the unit_code and the total is displayed (due to HAVING clause)

Subqueries and Correlated Queries:

- Query within a query
- The use of joins in a relational database allows you to get information from two or more tables
- It is often necessary to process data based on other processed data
- Characteristics:

• A subquery is a query (SELECT statement) inside another query.
• A subquery is normally expressed inside parentheses.
• The first query in the SQL statement is known as the outer query.
• The query inside the SQL statement is known as the inner query.
• The inner query is executed first.
• The output of an inner query is used as the input for the outer query.
• The entire SQL statement is sometimes referred to as a nested query.

- Subqueries can return the following:
    - One single value (one column and one row)
    - A list of values (one column and multiple rows)
    - A virtual tables (multicolumn, multi-row set of values)



- Returned by inner query:
- Returned by outer query:
- Comparison Operators for Subquery
    - Operator for single value comparison

      =, <, >
        - Equality
            - IN
        - Inequality
            - ALL, ANY combined with <, >, >=, <=

NEX

Week 10: Advanced SQL

Nested subquery

- Independent of outer query
- Executed once
- Inner executed first, just once
- Can be used in logical statements
- Example:
    - *Select studid, unitcode, mark*
      from enrolment
      where (unitcode, mark) IN (select unitcode, max(mark)
      from enrolment group by unitcode);

Correlated subquery

- Related to outer query and is considered to be evaluated for each row of the outer query
- Need to have equal condition in where clause, which relates to outer query
- Sub-query evaluated for each row
- Example
    - *Select studid, unitcode, mark*
      from enrolment e1
      where mark = (select max(mark)
      from enrolment e2
      where e2.unitcode = e1.unitcode)

Inline subquery

-

Views:

- Virtual table derived from one or more base tables
- Used to store subqueries in which you may want to access once completing other queries for a long query
- Syntax:
      CREATE OR REPLACE VIEW [view_name] AS
      SELECT … ;
- Views do not exist in DB, they are conceptual

- Add SQL functions WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table
- "OR REPLACE" means to replace previous view with new one

Self Join:

- Join a table with itself
- No join condition = cartesian product
    - Takes first row of first table and combines with all rows of second table
    - Second row of first table and combines with all rows of second table
    - Slide 15: selects name of both tables and rename it as manager

Full Outer Join

- Returns natural join output
- Any rows missing in tables are presented and columns are represented using nulls

Left Outer Join

- Returns natural join output
- Adds missing rows from left table and not right

Relational Set Operators

- Combine two or more sets to create a new one
- All have equal precedence (left to right)
- Two sets must be union compatible (same number of attributes and data types)
- Union
    - All rows selected by either query, incl. All duplicates
- Intersect
    - All distinct rows selected by both queries
- Union
    - All rows selected by either query, excl. duplicates
- Minus
    - All distinct rows selected by first query, not second

Week 11: Web database interface

Database Connectivity

- Data Layer
    - Data management application
- Database middleware
    - Manages connection between data transformation issues
        - Native SQL connectivity
        - Microsoft ODBC
        - Java Database Connectivity
- Application
    - External interface, mostly form of API