COMP810 Data Warehousing and Big Data

Semester 2 2024

Dr Victor Miranda

COMP810

Heads up: Week 4 Data Warehousing

- Database Diagram Design Tools
- SQL operations

- Database Diagram Design Tools

ERD (https://erdplus.com/)

ERD

A database modeling tool for creating Entity Relationship Diagrams, Relational Schemas, Star Schemas, and SQL DDL statements.

Database Modeling

ERDPlus is a web-based database modeling tool that lets you quickly and easily create

- Entity Relationship Diagrams (ERDs)
- Relational Schemas (Relational Diagrams)
- Star Schemas (Dimensional Models)

More features

- Automatically convert ER Diagrams into Relational Schemas
- Export SQL
- Export diagrams as a PNG
- Save diagrams safely on our server

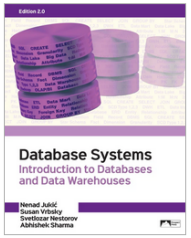
SQL DDL Statements

Export standard SQL

- Generate SQL from Relational Schemas and Star Schemas
- Select from common data types and data sizes
- Works with most contemporary RDBMS tools including Oracle, MySQL, Microsoft SQL Server, PostgresSQL, Teradata, IBM DB2, Microsoft Access, and others.

Textbook

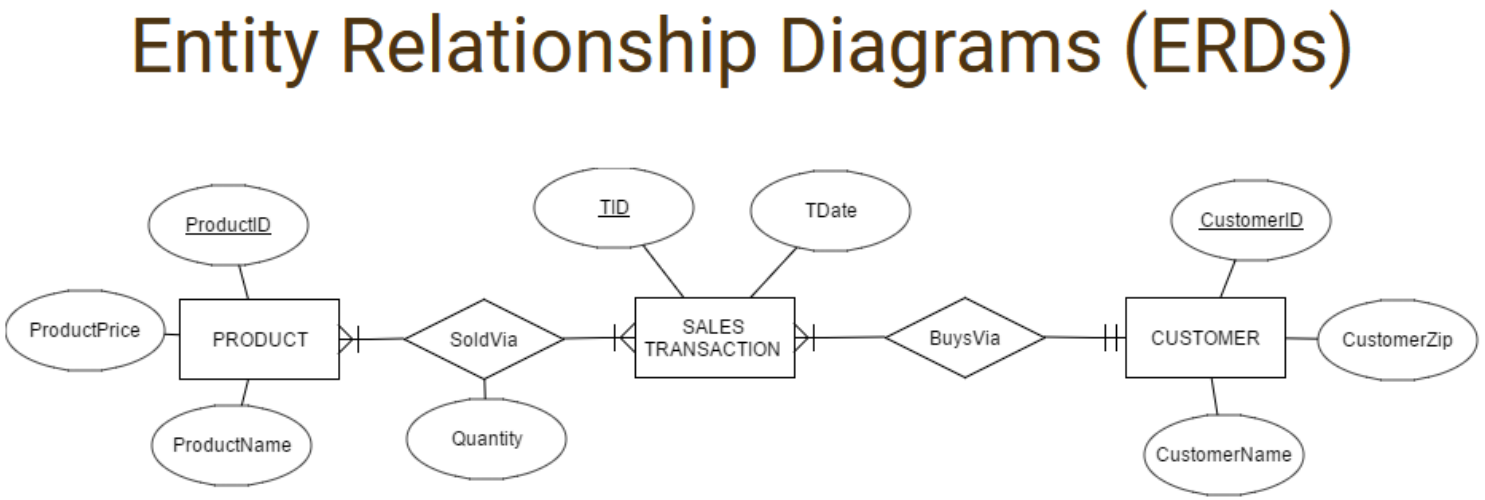
The Edition 2.0 of the textbook using ERDPlus is now available.



One of the most efficient ER Model tools that can be used to create

- ❑ Entity Relationship Diagrams (ERDs)
- ❑ Relational Schemas (Relational Diagrams)
- ❑ Dimensional Models – Star schema, snowflake etc., and
- ❑ SQL DDL statements

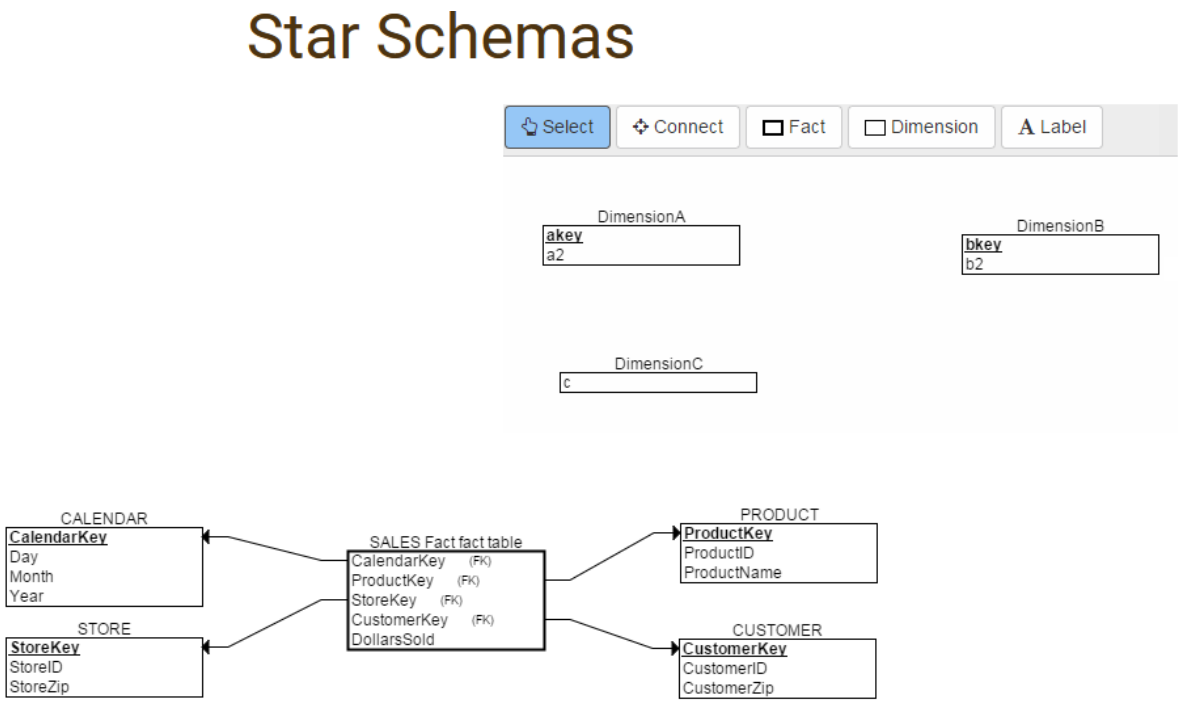
ERD (https://erdplus.com/)



ERDPlus enables drawing standard ERD components:

- Entities
- Attributes
- Relationships

ERD (https://erdplus.com/)



ERDPlus enables drawing dimensional models' components:

* Fact Tables and Dimension Tables
* Table Columns (including Primary and Foreign Keys)
* Referential Integrity Constraint Lines (pointing from a Foreign Key to the Primary Key it refers to)

ERD (https://erdplus.com/)

ERD

A database modeling tool for creating Entity Relationship Diagrams, Relational Schemas, Star Schemas, and SQL DDL statements.

Database Modeling

ERDPlus is a web-based database modeling tool that lets you quickly and easily create

- Entity Relationship Diagrams (ERDs)
- Relational Schemas (Relational Diagrams)
- Star Schemas (Dimensional Models)

More features

- Automatically convert ER Diagrams into Relational Schemas
- Export SQL
- Export diagrams as a PNG
- Save diagrams safely on our server

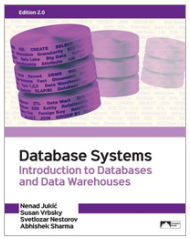
SQL DDL Statements

Export standard SQL

- Generate SQL from Relational Schemas and Star Schemas
- Select from common data types and data sizes
- Works with most contemporary RDBMS tools including Oracle, MySQL, Microsoft SQL Server, PostgresSQL, Teradata, IBM DB2, Microsoft Access, and others.

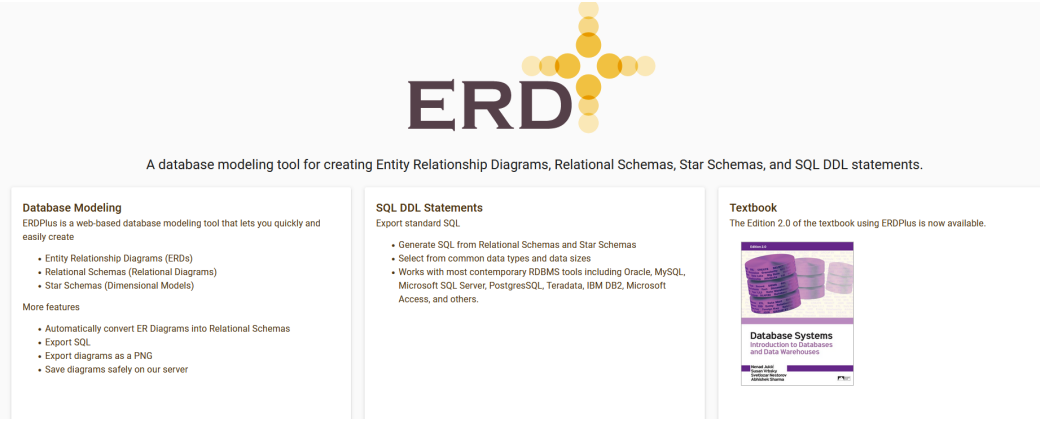
Textbook

The Edition 2.0 of the textbook using ERDPlus is now available.



DEMO…

ERD (https://erdplus.com/)

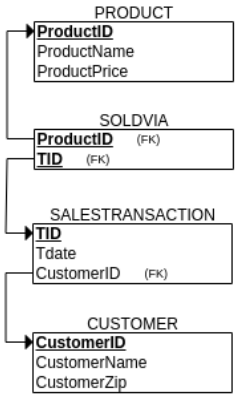


> Provides a visual starting point for database design.

> Availability of several features, to be used as fit

> FREE

ERD (https://erdplus.com/)



Export SQL

```
CREATE TABLE PRODUCT
(
  ProductID INT NOT NULL,
  ProductName VARCHAR(64) NOT NULL,
  ProductPrice DECIMAL(4, 2) NOT NULL,
  PRIMARY KEY (ProductID)
);

CREATE TABLE CUSTOMER
(
  CustomerID INT NOT NULL,
  CustomerName VARCHAR(40) NOT NULL,
  CustomerZip VARCHAR(10) NOT NULL,
  PRIMARY KEY (CustomerID)
);

CREATE TABLE SALESTRANSACTION
(
  TID INT NOT NULL,
  Tdate DATE NOT NULL,
  CustomerID INTEGER NOT NULL,
  PRIMARY KEY (TID),
  FOREIGN KEY (CustomerID) REFERENCES CUSTOMER(CustomerID)
);

CREATE TABLE SOLDVIA
(
  ProductID INT NOT NULL,
  TID INT NOT NULL,
  PRIMARY KEY (ProductID, TID),
  FOREIGN KEY (ProductID) REFERENCES PRODUCT(ProductID),
  FOREIGN KEY (TID) REFERENCES SALESTRANSACTION(TID)
);
```

https://datacadamia.com/data/type/relation/sql/decimal

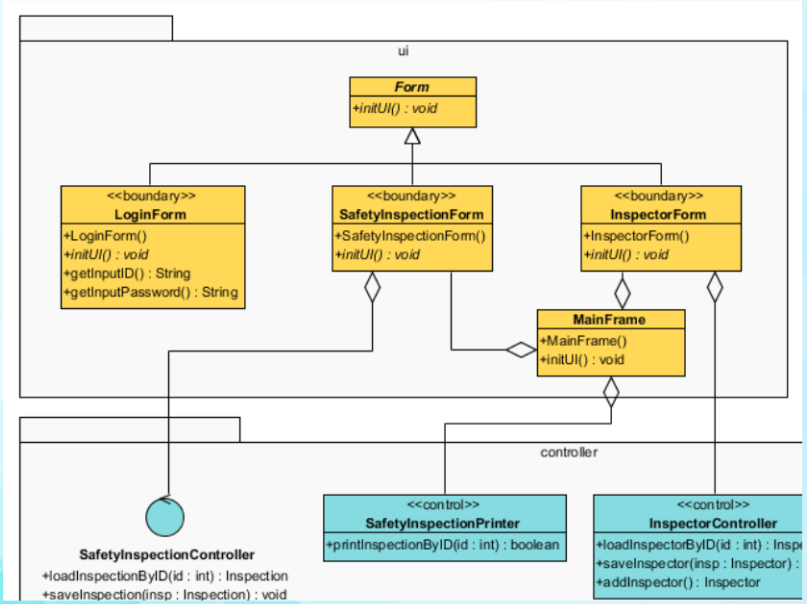
Visual Paradigm

Visual Paradigm What's New Features Tutorials Support Pricing Try Now Request Demo VP Online

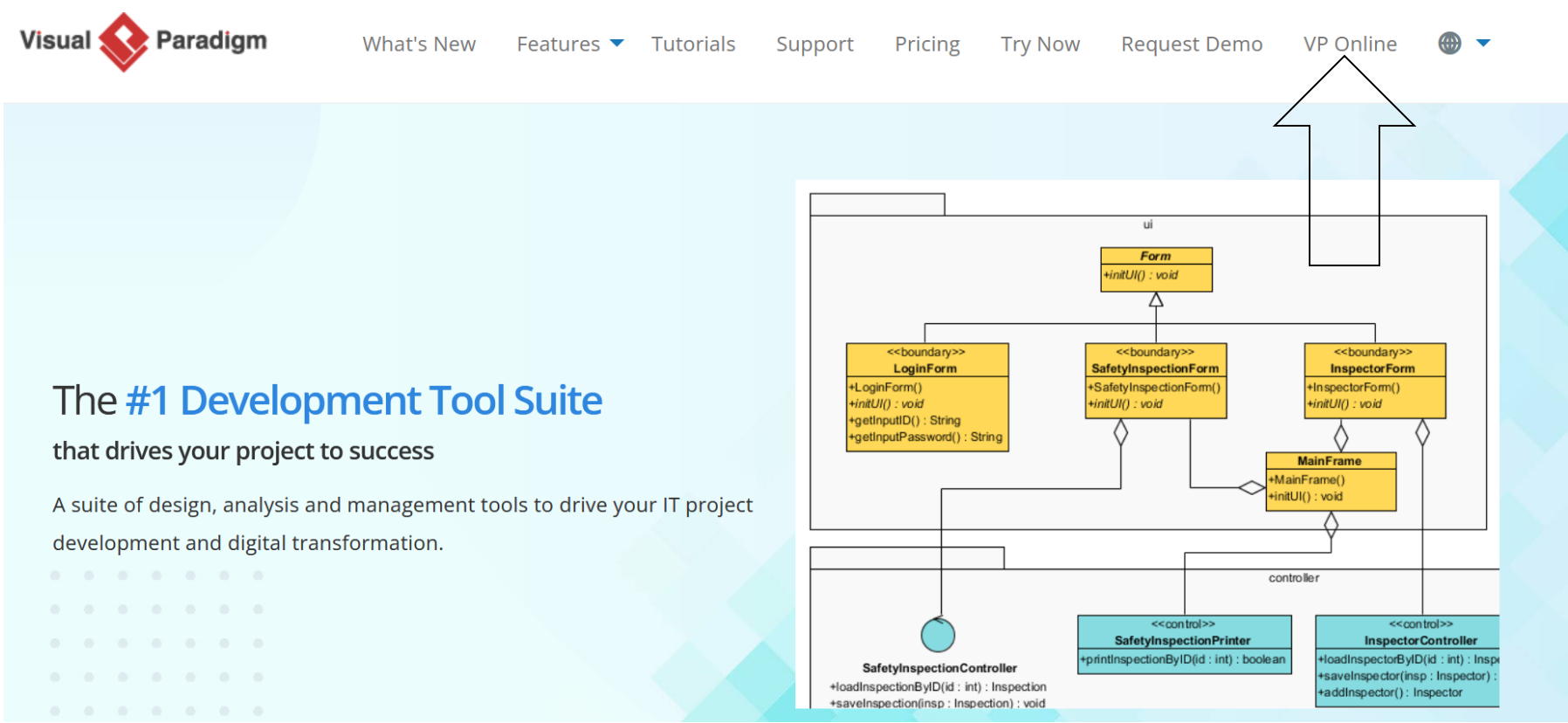
The #1 Development Tool Suite

that drives your project to success

A suite of design, analysis and management tools to drive your IT project development and digital transformation.

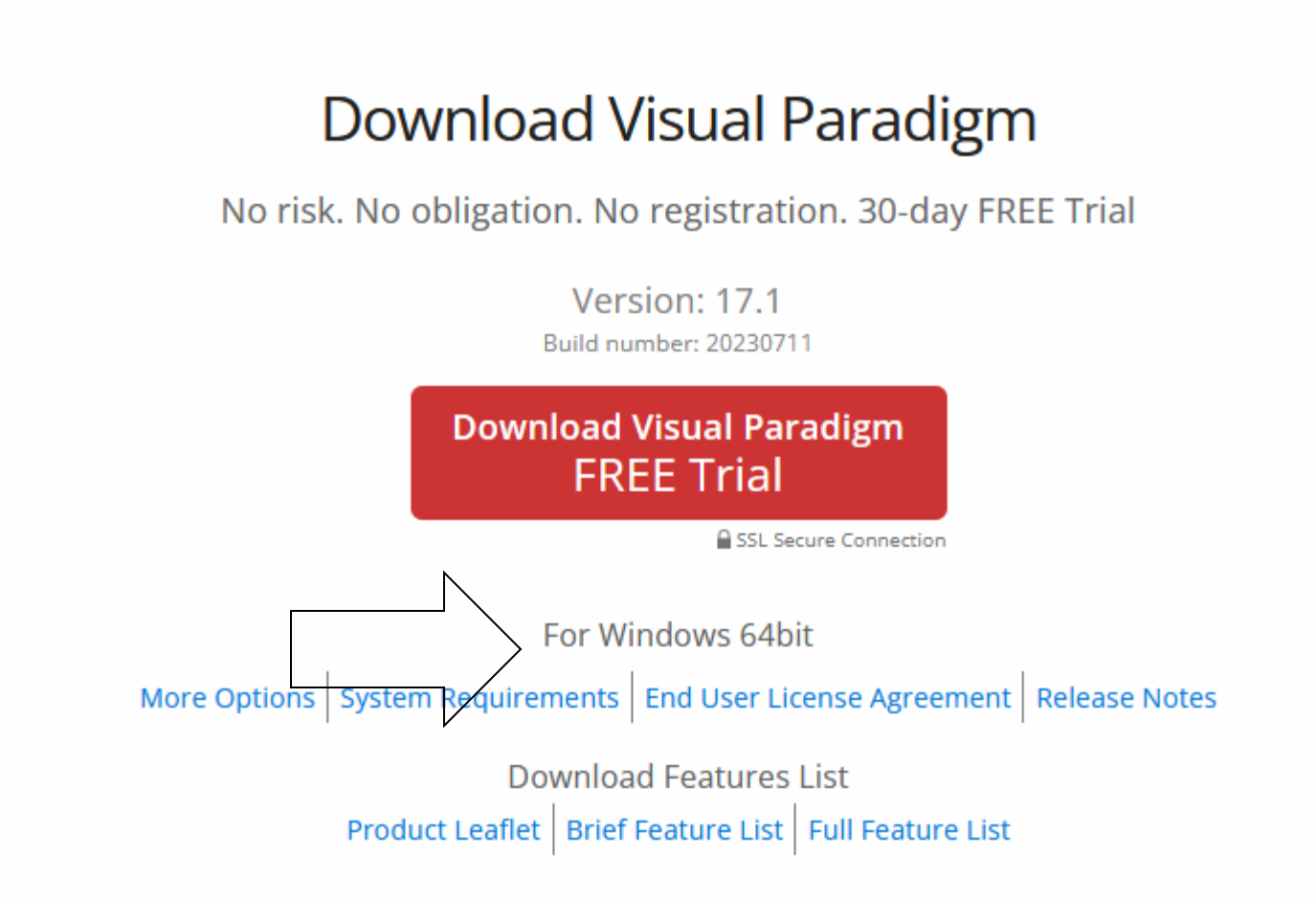


Visual Paradigm (online version – limited)

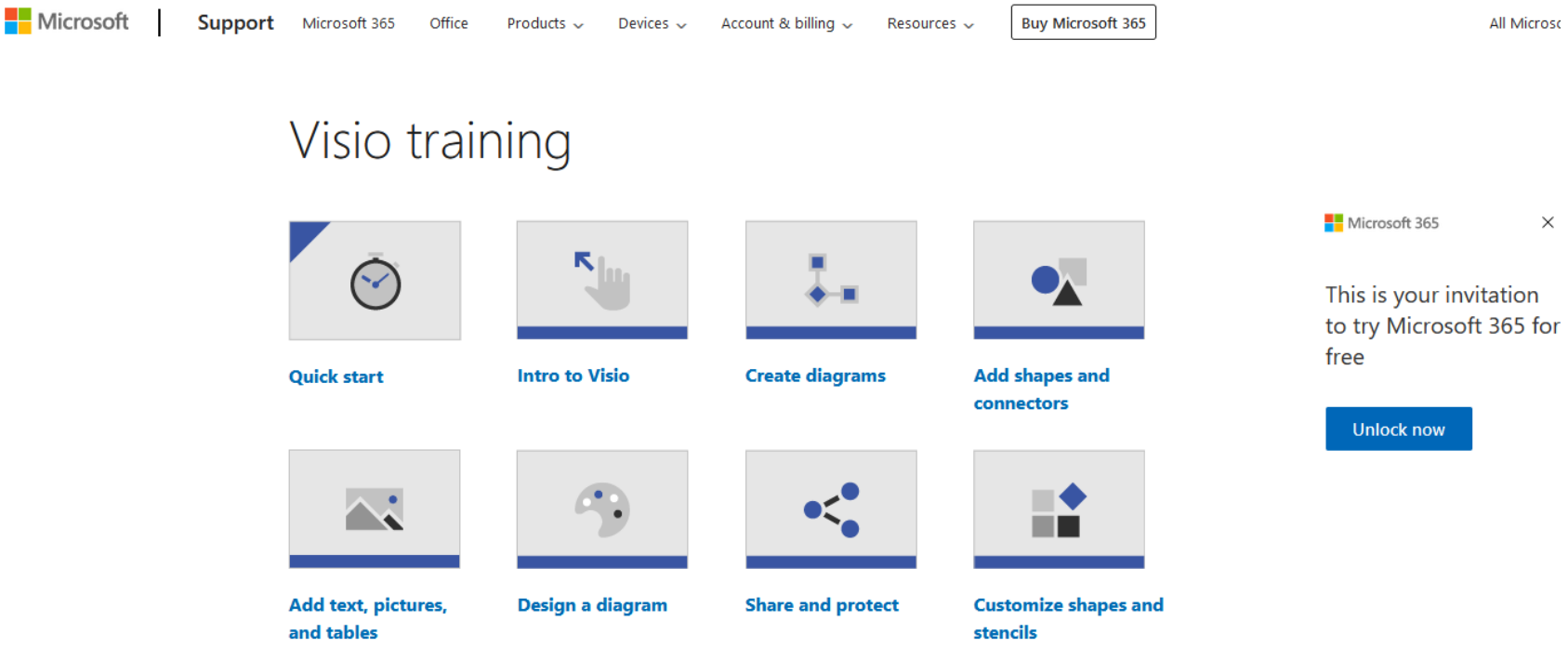


https://www.visual-paradigm.com/

Visual Paradigm (free trial)

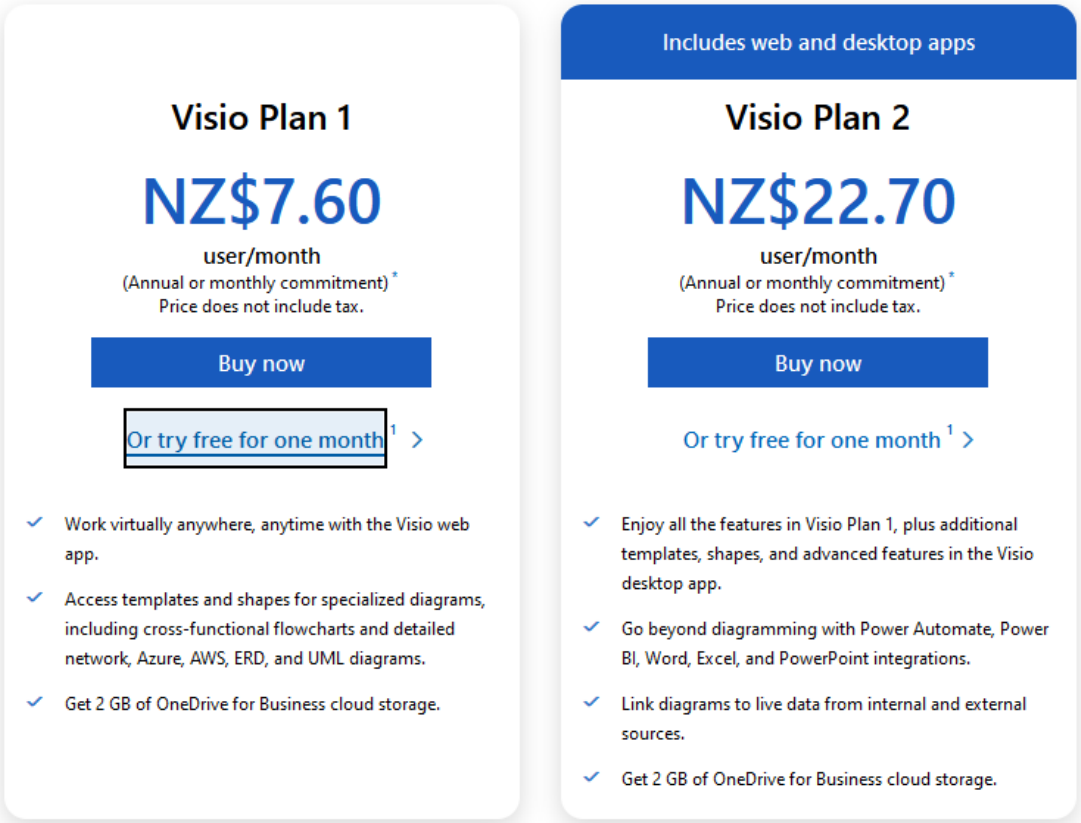


Visio



https://www.microsoft.com/en-nz/microsoft-365/visio/flowchart-software

Visio



https://www.microsoft.com/en-nz/microsoft-365/visio/flowchart-software

Different Options

Pick the option that suits your skills… and budget (pricing, features, etc.)

Keep it simple but self-contained and self-explanatory

Group by vs Nested Queries

1. Group by vs Nested Queries
2. Having Clause

Group by vs Nested Queries

	PRODUCT	PRICE	QUANTITY
1	A	4	14
2	B	6	13
3	A	4	31
4	B	6	29
5	C	8	11
6	C	8	4
7	B	6	7
8	C	8	8

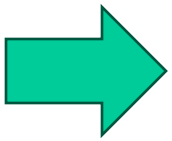
GROUP BY v.s. Nested Quereis

```
SELECT    product, Sum(price*quantity) AS TotalSales
FROM      Purchase
WHERE     date > '10/1/2005'
GROUP BY  product
```

GROUP BY v.s. Nested Quereis

```
SELECT    product, Sum(price*quantity) AS TotalSales
FROM      Purchase
WHERE     date > '10/1/2005'
GROUP BY  product
```

> 10/01/2005	Product	price	quantity
	A	4	14
	B	6	13
	A	4	31
	B	6	29
	C	8	11
	C	8	4
	B	6	7
	C	8	8



Product	Total_sales
A	180
B	294
C	184

GROUP BY v.s. Nested Quereis

```
SELECT x.product, (SELECT Sum(y.price*y.quantity)
                   FROM   Purchase y
                   WHERE  x.product = y.product
                          AND y.date > '10/1/2005')
                   AS TotalSales
FROM   Purchase x
WHERE  x.date > '10/1/2005';
```

GROUP BY v.s. Nested Quereis

```
SELECT x.product, (SELECT Sum(y.price*y.quantity)
                   FROM   Purchase y
                   WHERE  x.product = y.product
                          AND y.date > '10/1/2005')
                   AS TotalSales
FROM   Purchase x
WHERE  x.date > '10/1/2005';
```

	PRODUCT	TOTALSALES
1	B	294
2	B	294
3	B	294
4	C	184
5	C	184
6	C	184
7	A	180
8	A	180

GROUP BY v.s. Nested Quereis

```
SELECT DISTINCT  x.product, (SELECT Sum(y.price*y.quantity)
                             FROM     Purchase y
                             WHERE x.product = y.product
                                    AND y.date > '10/1/2005')
                            AS TotalSales
FROM         Purchase x
WHERE        x.date > '10/1/2005';
```

	PRODUCT	TOTALSALES
1	A	180
2	C	184
3	B	294

GROUP BY v.s. Nested Quereis

```
SELECT    product, Sum(price*quantity) AS TotalSales
FROM      Purchase
WHERE     date > '10/1/2005'
GROUP BY  product;
```

```
SELECT DISTINCT  x.product, (SELECT Sum(y.price*y.quantity)
                             FROM    Purchase y
                             WHERE x.product = y.product
                                   AND y.date > '10/1/2005')
                            AS TotalSales
FROM       Purchase x
WHERE      x.date > '10/1/2005';
```

Another Example

What does it mean ?

```
SELECT      product,
            sum(price * quantity) AS SumSales
            max(quantity) AS MaxQuantity
FROM        Purchase
GROUP BY product;
```

Another Example

```
SELECT      product,
            sum(price * quantity) AS SumSales
            max(quantity) AS MaxQuantity
FROM        Purchase
GROUP BY product;
```

	PRODUCT	SUMSALES	MAXQUANTITY
1	B	294	29
2	C	184	11
3	A	180	31

HAVING Clause

Same query, except that we consider only products that had at least 30 buyers.

```
SELECT      product, Sum(price * quantity)
FROM        Purchase
WHERE       date > '10/1/2005'
GROUP BY    product
HAVING      Sum(quantity) > 30;
```

HAVING clause contains conditions on aggregates.

HAVING Clause

```
SELECT      product, Sum(price * quantity)
FROM        Purchase
WHERE       date > '10/1/2005'
GROUP BY    product
HAVING      Sum(quantity) > 30;
```

	PRODUCT	SUMSALES
1	B	294
2	A	180

HAVING clause contains conditions on aggregates.

General form of Grouping and Aggregation

SELECT S
FROM $R_1,\ldots,R_n$
WHERE C1
GROUP BY $a_1,\ldots,a_k$
HAVING C2

S = may contain attributes $a_1,\ldots,a_k$ and/or any aggregates
C1 = is any condition on the attributes in $R_1,\ldots,R_n$
C2 = is any condition on aggregate expressions

General form of Grouping and Aggregation

```
SELECT    S
FROM      R1,…,Rn
WHERE     C1
GROUP BY  a1,…,ak
HAVING    C2
```

Evaluation steps:

1. Evaluate FROM-WHERE, apply condition C1
2. Group by the attributes $a_1,\ldots,a_k$
3. Apply condition C2 to each group (may have aggregates)
4. Compute aggregates in S and return the result

Slicing-and-Dicing

- When we use WHERE to specify a particular value for an axis (or several axes), we are performing a ***slice***
  - Slicing the data cube in the Time dimension (choosing sales only in week 12) then pivoting to *Product_id* (aggregating over *Market_id*)

```
SELECT   S.Product_Id,  SUM (Sales_Amt)
FROM     Sales S, Time T
WHERE    T.Time_Id = S.Time_Id  AND  T.Week = 'Wk-12'
GROUP BY  S. Product_Id;
```

Slice

Pivot

Slicing-and-Dicing

- Typically slicing and dicing involves several queries to find the "right slice."

  For instance, change the slice and the axes:

  - Slicing on Time and Market dimensions then pivoting to *Product_id* and *Week* (in the time dimension)

```
SELECT    S.Product_Id,  T.Quarter,  SUM (Sales_Amt)
FROM      Sales S,  Time T
WHERE     T.Time_Id = S.Time_Id
              AND  T.Quarter =  4
              AND  S.Market_id = 12345
GROUP BY  S.Product_Id,  T.Week;
```

Slice

Pivot

Example

- To construct the following table, we group the data twice..

Market_Id

SUM(*Sales_Amt*)	M1	M2	M3	*Total*
P1	3003	1503	…	…
P2	6003	2402	…	…
P3	4503	3	…	…
P4	7503	7000	…	…
Total	…	…	…	…

Product_Id

Total by row and column

- For the table entries, without the totals (aggregation on time)

```
SELECT    S.Market_Id, S.Product_Id, SUM
(S.Sales_Amt)
FROM      Sales S
GROUP BY  S.Market_Id, S.Product_Id;
```

Other options

- For the table entries, without the totals (aggregation on time)

```
SELECT    S.Market_Id, S.Product_Id, SUM (S.Sales_Amt)
FROM      Sales S
GROUP BY  S.Market_Id, S.Product_Id
```

- For the row totals (aggregation on time and supermarkets)

```
SELECT    S.Product_Id, SUM (S.Sales_Amt)
FROM      Sales S
GROUP BY  S.Product_Id
```

- For the column totals (aggregation on time and products)

```
SELECT    S.Market_Id, SUM (S.Sales)
FROM      Sales S
GROUP BY  S.Market_Id
```

References:

(a) A Conceptual Poverty Mapping Data Model
Link: https://www.researchgate.net/figure/Key-thematic-layers-for-poverty-spatial-data-modeling_fig2_229724703

(b) Relational Database relationships
https://www.youtube.com/watch?v=C3icLzBtg8I

(c) https://courses.ischool.berkeley.edu/i202/f97/Lecture13/DatabaseDesign/sld002.htm

(d) https://nexwebsites.com/database/database-management-systems/

(e) Acknowledgement – Thanks to http://courses.cs.washington.edu/courses/cse544/ for providing part of this presentation.

(f) Acknowledgement – Thanks to © Silberchatz, Korth and Surdashan for providing part of this presentation.

(e) Malinowski, Elzbieta, Zimányi, Esteban (2008) *Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications .* Springer Berlin Heidelberg. Copyright © 2008 Elzbieta Malinowski & Esteban Zimányi