ITCS393 Database Systems Lab

Database Views and Indexes

Dr. Petch Sajjacholapunt

Dr. Wudhichart Sawangphol

Dr. Jidapa Kraisangka

petch.saj@mahidol.ac.th

Learning Outcomes

After this class, students should be able to:

- explain the different between tables and views
- demonstrate how to manage (i.e., create, drop) views
- explain the benefit of index
- demonstrate how to manage (i.e., create, drop) index

Views

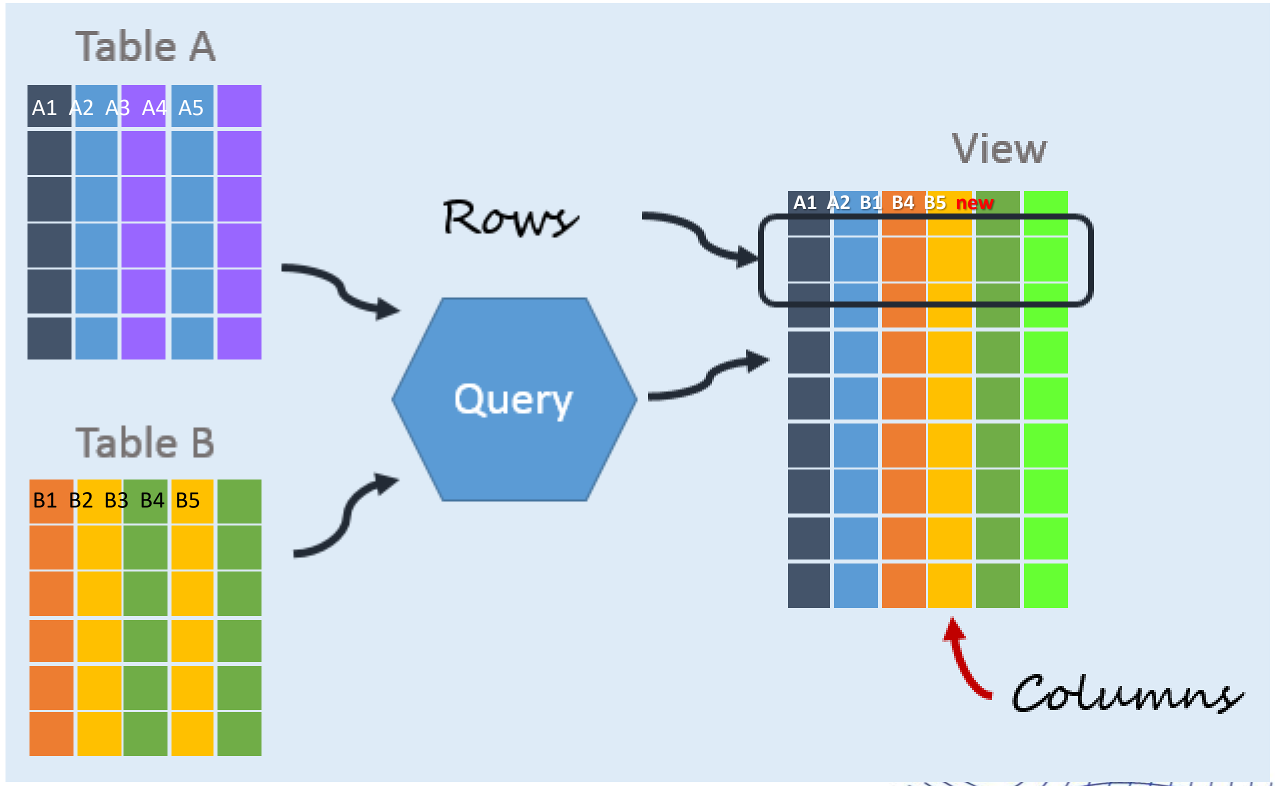
Views

- Lesson 1: Introduction to Database Views
- Lesson 2: Creating and Dropping Views
- Lesson 3: Selecting and Updating Data via Views
- Lesson 4: Managing Views

Lesson 1: Introduction to Database Views

View is a searchable object, or a *virtual* table defined by a SELECT query

- One or more source tables make up a view,
- Does not store data,
- Generally, read-only.



Modified from https://www.essentialsql.com/wp-content/uploads/2014/05/AnatomyOfAView.png

Advantage

- Hide complicated query
- Provide extra security layer to expose read-only data to specific users
- Consistency and enable computer/derived columns

Disadvantage

- Performance (need to rerun)
- Table dependency
- Cannot be associated with trigger

Syntax :

- Create

```
CREATE VIEW view_name
AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

- Alter

```
ALTER VIEW view_name
AS
SELECT column1, column2, ...
FROM table_name
WHERE new_condition;
```

- Drop

```
DROP VIEW view_name;
```

- Get information

```
SHOW CREATE VIEW view_name;
```

Lesson 1: Introduction to Database Views

Example:

- Create

```
CREATE VIEW recent_orders AS
SELECT order_id, customer_name,
order_date, total_amount
FROM orders
WHERE order_date >=
DATE_SUB(NOW(), INTERVAL 30 DAY);
```

- Alter

```
ALTER VIEW recent_orders AS
SELECT order_id, customer_name,
order_date, total_amount,
order_status
FROM orders
WHERE order_date >=
DATE_SUB(NOW(), INTERVAL 30 DAY);
```

- Drop

```
DROP VIEW recent_orders;
```

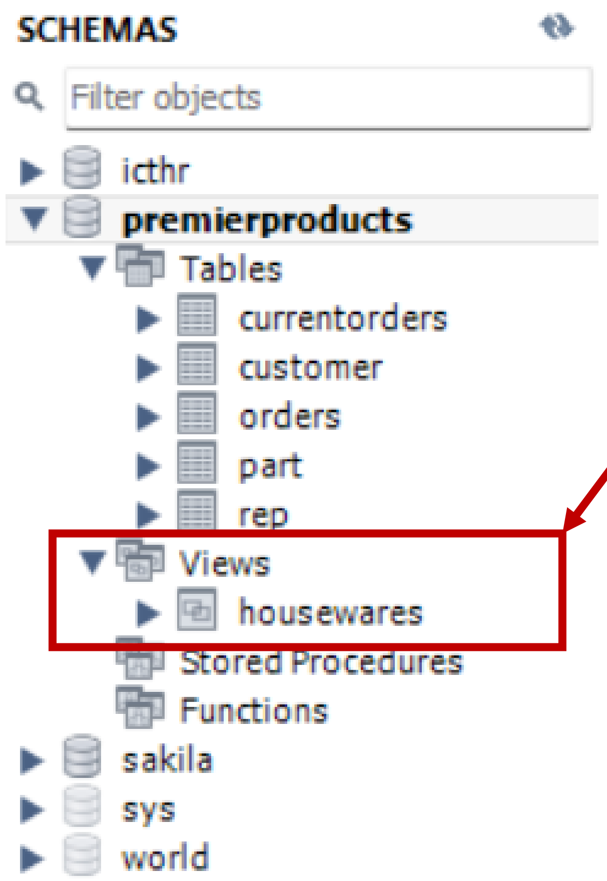
- Get information

```
SHOW CREATE VIEW recent_orders;
```

Lesson 2: Creating and Dropping Views

- Create *PremierProduct* database by execute *PremierProduct.sql*
- Execute *Data_PremierProducts.sql* to load data into the database.

2.1 Create and Drop View


SCHEMAS

Filter objects

- icthr
- premierproducts
  - Tables
    - currentorders
    - customer
    - orders
    - part
    - rep
  - Views
    - housewares
  - Stored Procedures
  - Functions
- sakila
- sys
- world

```
-- Q1 --
CREATE VIEW Housewares AS
SELECT PartNum, Description, OnHand, Price
FROM Part
WHERE Class = 'HW';
```

```
-- Q2 --
DROP VIEW Housewares;
```

2.2 Example of views

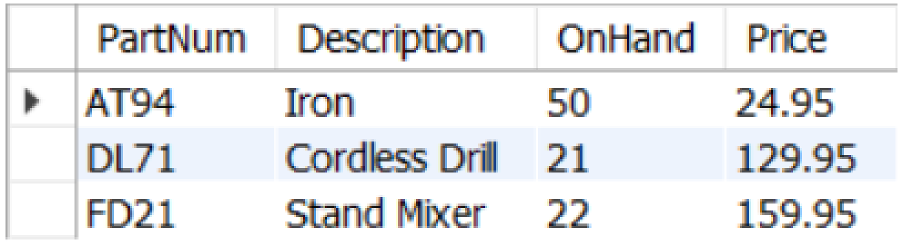
```
-- Q1.1 –  [Create view from multiple tables with alias name]

CREATE VIEW SalesRepCust (SNum, SLast, SFirst, CNum) AS

SELECT Rep.RepNum, LastName, FirstName, CustomerNum

FROM Rep, Customer

WHERE Rep.RepNum = Customer.RepNum;
```

```
-- Q1.2 – [Create view with derived attribute]

CREATE VIEW RepCountAvgBalance AS

SELECT RepNum, COUNT(*) AS NumCustomers, AVG(Balance) AS AvgBalance

FROM Customer

GROUP BY RepNum;
```

3.1 SELECT data using view VS. Table

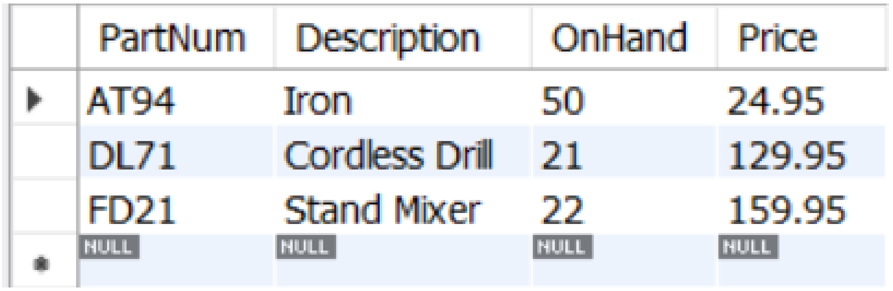
```
-- Q3 --
SELECT * FROM Housewares;

-- Q4 --
SELECT PartNum, Description, OnHand, Price
FROM Part
WHERE Class = 'HW';
```

Q3



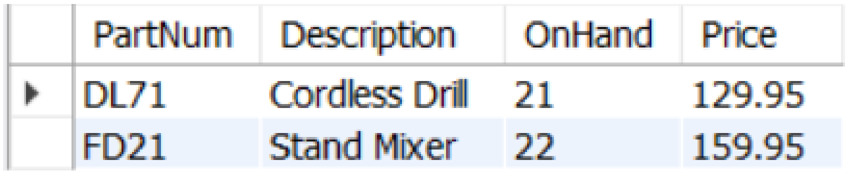
PartNum	Description	OnHand	Price
AT94	Iron	50	24.95
DL71	Cordless Drill	21	129.95
FD21	Stand Mixer	22	159.95

Q4



PartNum	Description	OnHand	Price
AT94	Iron	50	24.95
DL71	Cordless Drill	21	129.95
FD21	Stand Mixer	22	159.95
NULL	NULL	NULL	NULL

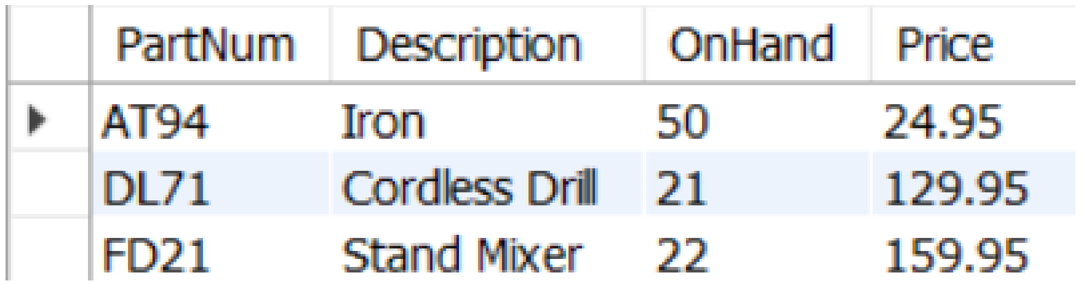
```
-- Q5 --
SELECT * FROM Housewares
WHERE OnHand < 25;
```



PartNum	Description	OnHand	Price
DL71	Cordless Drill	21	129.95
FD21	Stand Mixer	22	159.95

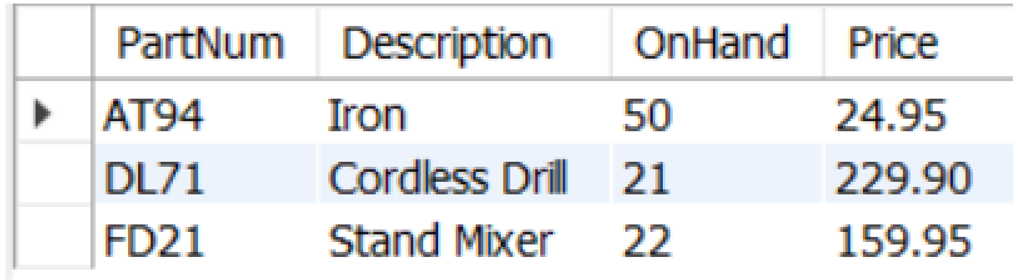
Can you write another query which does not use view?

3.2 UPDATE data using view VS. Table



PartNum	Description	OnHand	Price
AT94	Iron	50	24.95
DL71	Cordless Drill	21	129.95
FD21	Stand Mixer	22	159.95

```
-- Q7 --
UPDATE Housewares
SET Price = 229.90
WHERE PartNum = 'DL71';
```

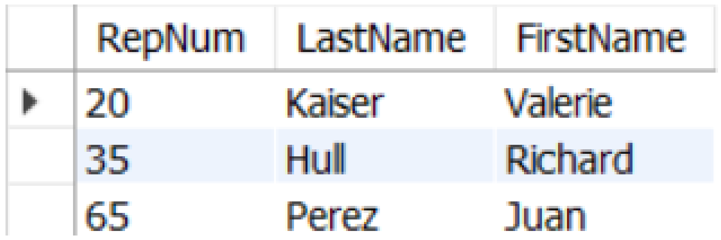


PartNum	Description	OnHand	Price
AT94	Iron	50	24.95
DL71	Cordless Drill	21	229.90
FD21	Stand Mixer	22	159.95

Can you write another query which does not use view?

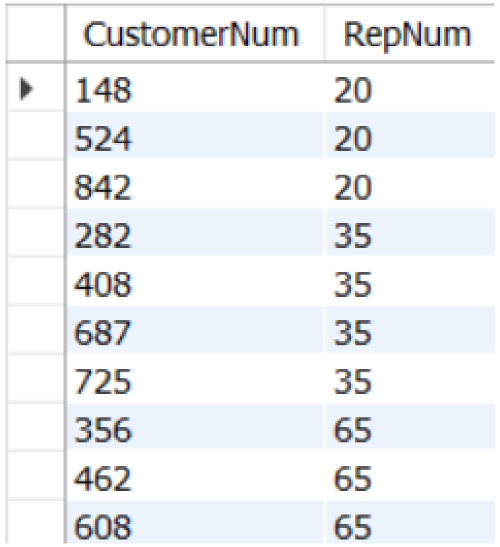
Create another view from multiple tables

```
SELECT RepNum, LastName, FirstName
FROM Rep;
```



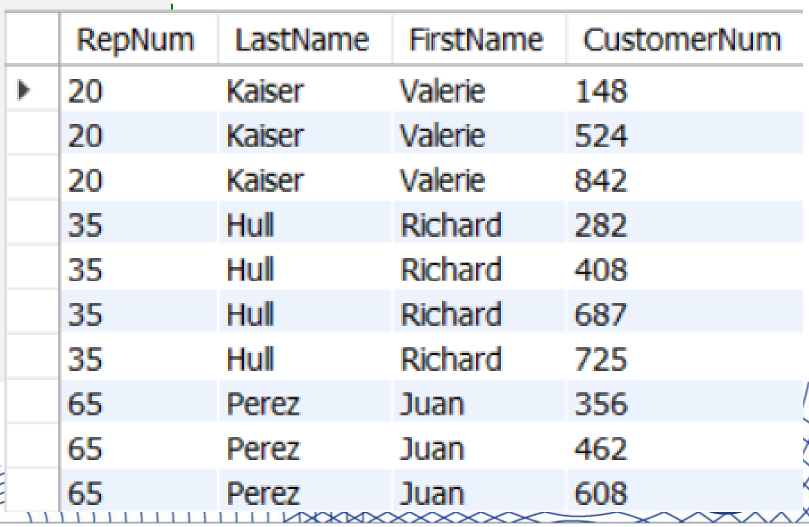
RepNum	LastName	FirstName
20	Kaiser	Valerie
35	Hull	Richard
65	Perez	Juan

```
SELECT CustomerNum, RepNum
FROM Customer;
```



CustomerNum	RepNum
148	20
524	20
842	20
282	35
408	35
687	35
725	35
356	65
462	65
608	65

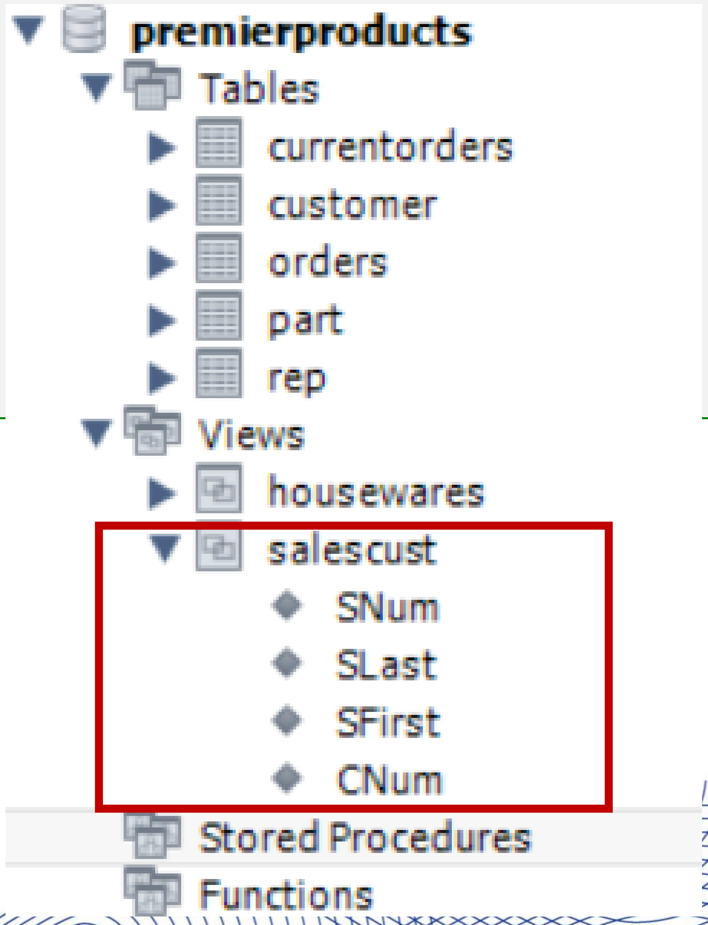
```
-- Join two tables

SELECT Rep.RepNum , LastName, FirstName, CustomerNum

FROM Rep, Customer

WHERE Rep.RepNum = Customer.RepNum;
```



RepNum	LastName	FirstName	CustomerNum
20	Kaiser	Valerie	148
20	Kaiser	Valerie	524
20	Kaiser	Valerie	842
35	Hull	Richard	282
35	Hull	Richard	408
35	Hull	Richard	687
35	Hull	Richard	725
65	Perez	Juan	356
65	Perez	Juan	462
65	Perez	Juan	608

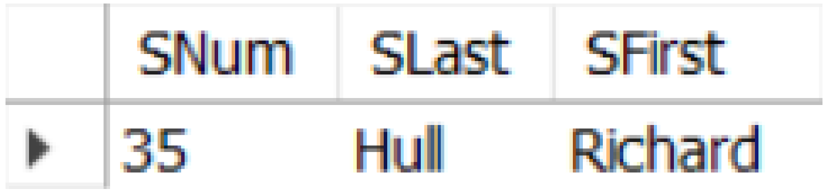
Create another view from multiple tables **with alias name**

```
-- Q9 – [Create view with alias name]

CREATE VIEW SalesCust (SNum, SLast, SFirst, CNum) AS

SELECT Rep.RepNum, LastName, FirstName, CustomerNum

FROM Rep, Customer

WHERE Rep.RepNum = Customer.RepNum;
```



3.3 SELECT data using view **with alias name**

```
-- Q10 --
SELECT SNum, SLast, SFirst
FROM SalesCust
WHERE CNum = '282';
```

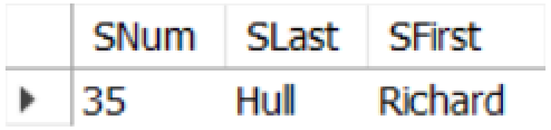


SNum	SLast	SFirst
35	Hull	Richard

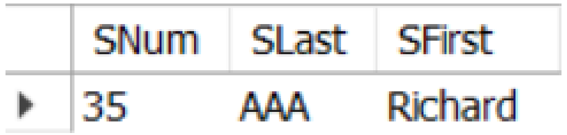
Can you write another query which does not use view?

3.3 UPDATE data via view – checking referential integrity

```sql
-- Q12 --
UPDATE SalesCust
SET SLast = 'AAA'
WHERE CNum = '282';
```



	SNum	SLast	SFirst
▶	35	Hull	Richard

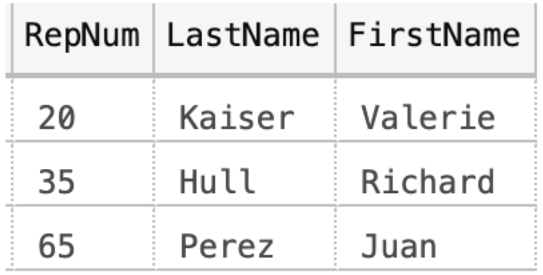


	SNum	SLast	SFirst
▶	35	AAA	Richard

```sql
-- Q13 -- INVALID FK
UPDATE SalesCust
SET SNum = '70'
WHERE CNum = '282';
```

```
Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`premierproducts`.`customer`, CONSTRAINT `customer_ibfk_1` FOREIGN KEY (`RepNum`) REFERENCES `rep` (`RepNum`))
```

Why Error?
SNum in the View represents **RepNum** in Rep Table.
Since, there is NO RepNum with value 70 in the Rep Table,
you cannot set SNum = '70' **[Violate referential integrity]**

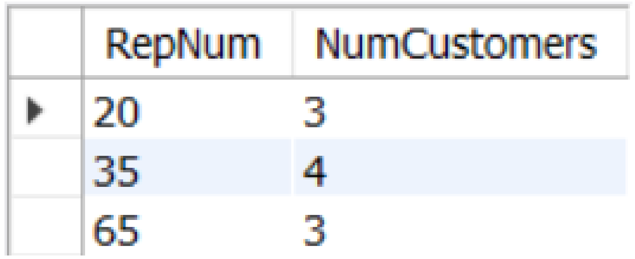


RepNum	LastName	FirstName
20	Kaiser	Valerie
35	Hull	Richard
65	Perez	Juan

3.3 CREATE, SELECT, and UPDATE view **with derived field**

```
-- Q14 --
CREATE VIEW CustCountPerRep AS
SELECT RepNum, COUNT(*) AS NumCustomers
FROM Customer
GROUP BY RepNum;
```

```
-- Q15 --
SELECT *
FROM CustCountPerRep;
```



RepNum	NumCustomers
20	3
35	4
65	3

```
-- Q16 --
UPDATE CustCountPerRep
SET RepNum = 10;
```

Cannot update the view or function 'CustCountPerRep' because it contains aggregates, or a DISTINCT or GROUP BY clause, or PIVOT or UNPIVOT operator.

Lesson 4: Managing Views

4.1 Alter View

Assume that the **Housewares** view was created from the following query.

```
CREATE VIEW Housewares AS
SELECT PartNum, Description, OnHand, Price
FROM Part
WHERE Class = 'HW';
```

```
SELECT * FROM Housewares;
```

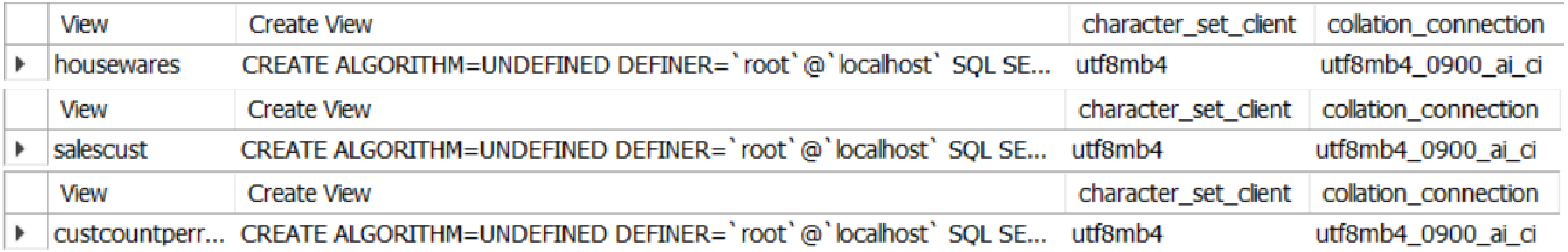
PartNum	Description	OnHand	Price
AT94	Iron	50	24.9500
DL71	Cordless Drill	21	229.9000
FD21	Stand Mixer	22	159.9500

```
-- Q17 --
ALTER VIEW Housewares AS
SELECT PartNum, Description, OnHand, Price
FROM Part
WHERE Class = 'HW' AND OnHand < 25;
```

PartNum	Description	OnHand	Price
DL71	Cordless Drill	21	229.9000
FD21	Stand Mixer	22	159.9500

4.1 Auditing View using Stored Procedure “sp_helptext”

```
-- Q18 --
SHOW CREATE TABLE Housewares;
SHOW CREATE TABLE SalesCust;
SHOW CREATE TABLE CustCountPerRep;
```



View	Create View	character_set_client	collation_connection
housewares	CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SE...	utf8mb4	utf8mb4_0900_ai_ci

View	Create View	character_set_client	collation_connection
salescust	CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SE...	utf8mb4	utf8mb4_0900_ai_ci

View	Create View	character_set_client	collation_connection
custcountperr...	CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SE...	utf8mb4	utf8mb4_0900_ai_ci

Index

Index

An **index** is a data structure that is used to speed up queries by allowing fast access to rows in a table that match a certain condition.

Indexes are used to find rows with specific column values quickly. **Without an index**, MySQL must begin with the first row and then read through the entire table to find the relevant rows. The larger the table, the more this costs.

If the table has an index for the columns in question, MySQL can quickly determine the position to seek to in the middle of the data file without having to look at all the data. This is much faster than reading every row sequentially.

Lesson 1: Intro to Database Index

- Re-Create PremierProduct database.
- Create the following tables

```sql
-- Create a DataStore Table
CREATE TABLE DataStore (
     Run INT AUTO_INCREMENT NOT NULL,
     KeyID INT NOT NULL,
     AccountDesc NVARCHAR(50),
     AccountType NVARCHAR(50),
     CodeAltKey INT,
     PRIMARY KEY (Run)
);


-- Create a DataStore_IDX Table
CREATE TABLE DataStore_IDX (
     Run INT AUTO_INCREMENT NOT NULL,
     KeyID INT NOT NULL,
     AccountDesc NVARCHAR(50),
     AccountType NVARCHAR(50),
     CodeAltKey INT,
     PRIMARY KEY (Run)

);
```

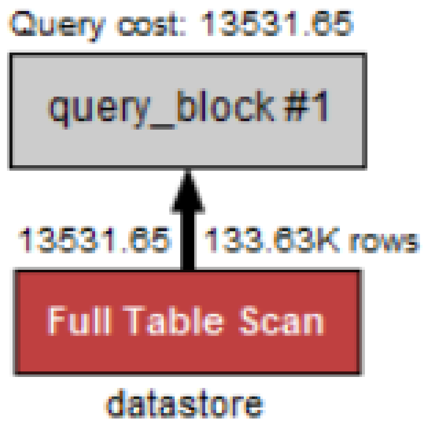
1.0 Prepare Large Tables

```sql
-- Generate data and insert into the DataStore and DataStore_IDX table
INSERT INTO `DataStore` (`KeyID`, `AccountDesc`, `AccountType`, `CodeAltKey`)
SELECT FLOOR(RAND() * 10000000),
       CONCAT('accountdesc', CAST(FLOOR(RAND() * 10000000) AS CHAR)),
       CONCAT('AccountType', CAST(FLOOR(RAND() * 10000000) AS CHAR)),
       FLOOR(RAND() * 10000000)
FROM `information_schema`.`tables` AS t1
CROSS JOIN `information_schema`.`tables` AS t2
LIMIT 2700000;

INSERT INTO `DataStore_IDX` (`KeyID`, `AccountDesc`, `AccountType`, `CodeAltKey`)
SELECT `KeyID`, `AccountDesc`, `AccountType`, `CodeAltKey`
FROM `DataStore`;
```

1.1 How data can be accessed in SQL Server

- Using ***a table scan***
  - Look through every row to determine if any records meet the conditions.

select * from datastore where AccountType = "AccountType6591660";

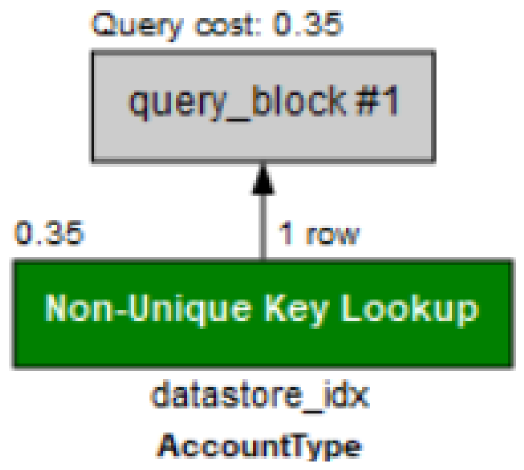


Let try to create single colum index:

```
ALTER TABLE datastore_idx ADD INDEX (AccountType)
```

Result:

select * from datastore_idx where AccountType = "AccountType6591660";

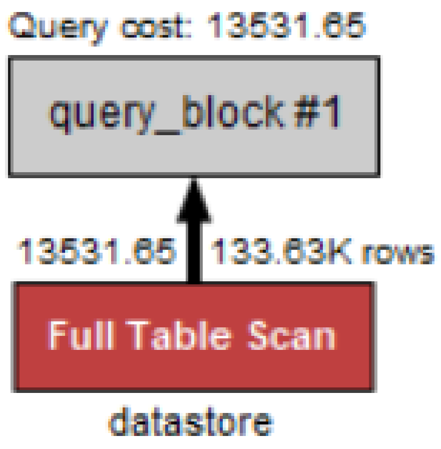


Lesson 1: Intro to Database Index

Like many relational database engines, MySQL allows you to create indexes that are composed of multiple columns:

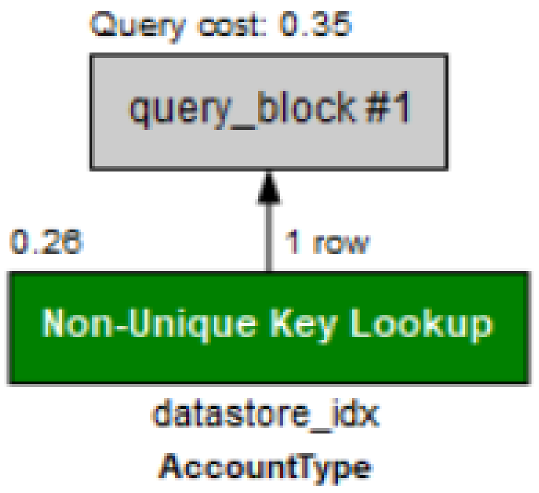
```
ALTER TABLE datastore_idx ADD INDEX (AccountType, AccountDesc)
```

```
SELECT * FROM datastore
WHERE AccountDesc = 'accountdesc8804227’
AND AccountType = 'AccountType8909795';
```



Like many relational database engines, MySQL allows you to create indexes that are composed of multiple columns:

```
SELECT * FROM datastore_idx
WHERE AccountDesc = 'accountdesc8804227'
AND AccountType = 'AccountType8909795';
```



- Tables that **should have** index:
  - Large tables
  - Tables that are frequently retrieved for a set of queries.

- Attributes that should be indexed:
  - Attributes used for joining (JOIN conditions)
  - Attributes used for sorting (in ORDER BY clause)
  - Attributes used for grouping (in GROUP BY clause)
  - Attributes used in aggregation functions
  - Attributes used in a WHERE clause
  - Attributes used as a foreign key

- Tables that **should NOT** have index:
  - Small tables
  - Tables that are frequently manipulated for a set of queries.
- Attributes that **should NOT** be indexed
  - Attributes with type image, bit and text
  - Attributes with small domain (e.g., gender)
  - Attributes with large size (e.g., char(100))
  - Attributes that are rarely used in any query

Mahidol University

THANKS
FOR YOUR
ATTENTION
ANY
QUESTIONS?