

Datenbanken 2

Oliver Lindemann
Hochschule Harz
University of Applied Science
u33873, m26264
u33873@hs-harz.de

ABSTRACT

This paper will introduce you to relational database systems and Data-Warehouses. It explains what they are used for and uses an Event-Shop to demonstrate how they are built.

1. EINLEITUNG

Datenbanken finden sich heutzutage in fast allen Anwendungsbe-
reichen wieder. Sei es die Kundenverwaltung eines Unternehmens,
der Warenbestand in einem Supermarkt oder die bevorzugte Such-
maschine. Überall werden Datenbanken zum Verwalten und Orga-
nisieren der Mengen an Daten eingesetzt. In dieser Facharbeit wer-
den relationale Datenbanksysteme (RDBS) vorgestellt, die Anwen-
dungsmöglichkeiten von Data-Warehouses und die Tabellenstruk-
tur des Event-Shop-Empfehlungssystems erläutert.

2. GRUNDLAGEN

Eine relationale Datenbank ist eine Ansammlung strukturierter Da-
ten, die miteinander in Relation stehen (können). Diese Daten wer-
den in Tabellen organisiert, welche die Datenstruktur abbilden.
Eine Tabelle besteht aus mehreren Spalten, welche jeweils eine Ei-
genschaft der Daten abbildet, und den Reihen, die jeweils einen
vollständigen Datensatz darstellen. Die Besonderheit einer relation-
alen Datenbank sind die Relationen / Beziehungen zwischen den
Daten bzw. den Tabellen. Diese erlauben das Referenzieren von
Datensätze anderer Tabellen in einem Datensatz¹. Allgemein ge-
sprochen können durch Relationen Redundanzen vermieden wer-
den, da auf einen einmal eingefügten Datensatz von anderen Stellen
verwiesen werden kann und dieser nicht an jeder dieser Stellen er-
neut eingefügt werden muss. Dieses Prinzip der Redundanzvermei-
dung wird *Normalisierung* genannt. Damit eine Datenbank *norma-*
lisiert ist, muss diese die drei aufeinander aufbauenden *Normalfor-*
men (NF) erfüllen:

- 1. **NF:** Alle Attribute sind atomar (kleinste mögliche Einheit)
- 2. **NF:** Alle Nichtschlüsselattribute sind vom Primärschlüssel abhängig
- 3. **NF:** Alle Nichtschlüsselattribute sind voneinander unabhän-
gig

Normalisierte und somit auch redundanzfreie Daten haben den
Vorteil, dass bei einer Änderung eines Datensatzes nur eine einzige
Stelle aktualisiert werden muss.

Modul: Datenbanken 2
Prüfer: Prof. Dr. Kerstin Schneider
Abgabedatum: 28.02.2020

Der Nachteil von normalisierten Daten ist allerdings die Geschwin-
digkeitseinbuße im operativen Betrieb. Einen vollständigen Daten-
satz mit all seinen Referenzen zu sammeln ist aufwändig, da die
einzelnen Datensätze über mehrere Tabellen verteilt liegen. Es
muss also über jede dieser Tabellen iteriert und nach den passenden
Daten gesucht werden, um einen vollständigen Datensatz zu erhal-
ten. Aus diesem Grund existieren andere Datenbankansätze, die be-
wusst gegen die Normalisierung verstoßen, um die Laufzeit zu ver-
bessern. Diese Systeme sind *denormalisiert*. Ein typisches Beispiel
eines solchen Systems ist das Data-Warehouse (DWH).

Eine Datenbank ist *denormalisiert*, sobald diese gegen eine der drei
Normalformen verstößt, beispielsweise bei Einführung bewusster
Redundanzen.

3. DATA-WAREHOUSE

Ein Data-Warehouse (DWH) ist *denormalisiert* und bildet in den
meisten Fällen bereits existierende Daten von einer oder mehreren
normalisierten Datenbanken ab. Die Daten werden i.d.R. noch ver-
arbeitet und erweitert, indem Ableitungen von diesen erstellt und
ebenfalls in den Tabellen gespeichert werden. Dies nimmt eventu-
elle Berechnung von Abfragen und Analysen vorweg und verbes-
sert somit die Geschwindigkeit. Aus diesem Grund werden in das
DWH vom Anwender keine neuen Daten eingefügt, sondern ledig-
lich auf die in den operationalen Systemen verfügbaren Daten zu-
gegriffen. Dem Data-Warehouse liegen also in den meisten Fällen,
wie in *Abbildung 1 Data-Warehouse Architektur* dargestellt, bereits
ein oder mehrere bestehende Datenbanken zugrunde.

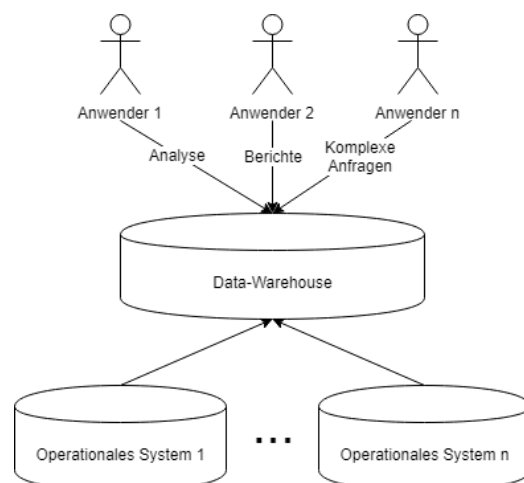


Abbildung 1 Data-Warehouse Architektur

¹ Beispielsweise referenziert eine Bewertung ein spezielles Event,
für das diese Bewertung geschrieben wurde.

Der Unterschied zwischen der *Normalisierung* und der *Denormalisierung* lässt sich anhand eines Datums in einer Tabelle darstellen:

In einer *normalisierten* Tabelle liegt nur das einzelne Datum ab. Aus diesem Datum lassen sich später Tag, Monat, Jahr und noch weitere Daten ableiten.

In einer *denormalisierten* Tabelle liegen neben dem Datum nun auch noch die abgeleiteten Eigenschaften des Datums ab, wie der Tag, der Monat, das Jahr, die Kalenderwoche und vieles. Dies erspart für zukünftige Abfragen und Analysen die Berechnung der abgeleiteten Daten und erhöht die Geschwindigkeit des DWH.

3.1 DWH Schemas

Für die Realisierung von Data-Warehouses gibt es verschiedene Ansätze und damit einhergehend unterschiedliche Schemas [1]:

- Star Schema
- Snowflake Schema

Die Namen der Schemas ergeben sich aus der Anordnung der Tabellen und der damit entstehenden Figuren / Objekte.

3.1.1 Star Schema

Das Star-Schema ist ein einfaches Schema, welches sternförmig aufgebaut ist. Es besteht aus einer Faktentabelle, welche den Mittelpunkt des Sternes symbolisiert, und mehreren Dimensionstabellen, die sternförmig um Faktentabelle positioniert sind.

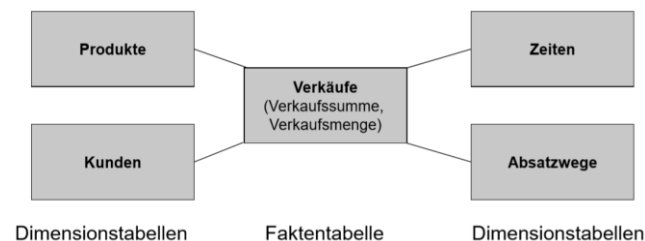


Abbildung 2 DWH Star Schema

Die Dimensionstabellen in einem Star Schema sind nicht in der 3. Normalform, sondern lediglich in der 2. Dies hat den Vorteil, dass alle relevanten Daten von der Faktentabelle mittels nur eines *Joins*² erreicht und gesammelt werden können.

3.1.2 Snowflake Schema

Das Snowflake Schema ist ein komplexeres DWH Schema. Dimensionstabellen liegen in der 3. Normalform vor und können, um Redundanzen zu vermeiden, in kleinere Tabellen unterteilt sein. Dies bedeutet auch, dass Dimensionstabellen mit weiteren Dimensionstabellen in Relation stehen können. Allerdings werden dadurch bei einer Abfrage auch mehrere *Joins* benötigt, um diese zu vereinen.

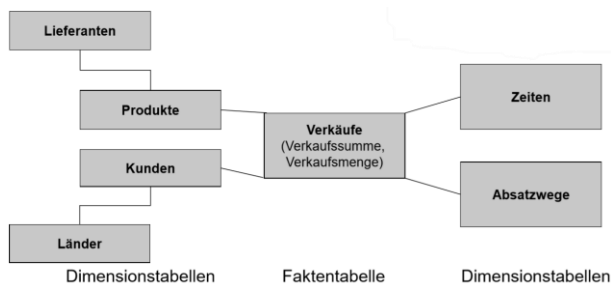


Abbildung 3 DWH Snowflake Schema

Faktentabelle: speichert die zu verwaltenden bzw. die zu analysierenden Daten, z.B. Ticketverkäufe.

Dimensionstabelle: speichert die Attribute und Eigenschaften von Fakten (bzw. beschreiben Fakten im Detail).

Faktentabellen enthalten Primärschlüssel der Dimensionstabellen.

4. EVENT-SHOP DATENBANK

Die Event-Shop Datenbank ist nach dem Data-Warehouse Prinzip aufgebaut. Allerdings fielen aus Gründen der Einfachheit die unterliegenden operationalen Datenbanken weg. Das DWH dient somit zugleich als operatives System.

Der Event-Shop hat folgende Anforderungen an die Datenbank:

- Ein Kunde (Customer) kann sich registrieren und mittels seiner E-Mail-Adresse und Passwortes anmelden.
- Der Kunde kann Veranstaltungen (Events) auswählen und kaufen.
- Der Kauf (Purchase) erfordert eine Anzahl an Tickets, eine Bezahlmethode (Paymethod) und eine Ticketart (Tickettype), die etwaigen Rabatt ermöglicht.
- Gekaufte Events kann der Kunde bewerten (Rating).

Aus diesen Anforderungen lassen sich verschiedene Tabellen entwickeln, die im weiteren Verlauf dieser Facharbeit erläutert werden.

4.1 Tabellen des Event-Shops

4.1.1 Events

Die Event-Tabelle beinhaltet alle verfügbaren Events. Diese Events sind gegliedert in Titel (TITLE), Beschreibung (DESCRIPTION), Typ (TYPENAME), Genre (GENRE) und Sub-Genre (SUB_GENRE). Zudem kommt der Veranstaltungsort mit der Straße (STREET), Hausnummer (HOUSENUMBER), Postleitzahl (ZIP), dem Ort (CITY) und Land (COUNTRY) inkl. Länderkürzel (COUNTRY_SHORT). Des Weiteren wird der Veranstalter (ORGANIZERNAME), das Veranstaltungsdatum (DATE) inkl. julianischer Datumsangabe (DATE_JULIAN) und die Dauer (DURATION_IN_MINUTES) des Events in Minuten angegeben. Zuletzt hat jedes Event noch einen Eintrittspreis (PRICE), der noch einmal als Centbetrag angegeben wird (PRICE_CENT).

LABOR_HARZ_007.EVENTS		
P *	EVENTID	NUMBER
*	TYPENAME	VARCHAR2 (20 BYTE)
	GENRE	VARCHAR2 (50 BYTE)
	SUB_GENRE	VARCHAR2 (50 BYTE)
*	STREET	VARCHAR2 (100 BYTE)
*	HOUSENUMBER	VARCHAR2 (4 BYTE)
*	ZIP	VARCHAR2 (6 BYTE)
*	CITY	VARCHAR2 (80 BYTE)
	COUNTRY	VARCHAR2 (60 BYTE)
*	COUNTRY_SHORT	CHAR (2 BYTE)
	ORGANIZERNAME	VARCHAR2 (60 BYTE)
	TITLE	VARCHAR2 (100 BYTE)
	DESCRIPTION	VARCHAR2 (500 BYTE)
	DATE	DATE
	DATE_JULIAN	NUMBER (10)
	DURATION_IN_MINUTES	NUMBER (4)
	PRICE	NUMBER (10,2)
	PRICE_CENT	NUMBER (10)
EVENTS_PK (EVENTID)		

Abbildung 4 Tabelle EVENTS

² Ein *Join* verknüpft / vereint zwei in Relation stehende Tabellen

4.1.2 Customer

Die Tabelle Customer beinhaltet alle registrierten Kunden. Die Spalten der Tabelle enthalten den Namen (FIRSTNAME, LASTNAME), das Geschlecht (GENDER, GENDER_SHORT), ihr Geburtsdatum als Datum (BIRTHDAY_DATE) und in julianischem Datumsformat (BIRTHDAY_JULIAN), das Geburtsjahr (BIRTHDAY_YEAR), die vollständige Adresse (STREET, HOUSENUMBER, ZIP, CITY, COUNTRY, COUNTRY_SHORT), die E-Mail-Adresse (EMAILADDRESS) und Provider (EMAIL_PROVIDER), ihr gewähltes Passwort (PASSWORD) und die ausgewählten Präferenzen (PREFERENCE1, PREFERENCE2, PREFERENCE3). Abbildung 5 stellt alle Spalten detailliert dar:



LABOR_HARZ_007.CUSTOMER		
P *	USERID	NUMBER
*	FIRSTNAME	VARCHAR2 (30 BYTE)
*	LASTNAME	VARCHAR2 (20 BYTE)
*	STREET	VARCHAR2 (100 BYTE)
*	HOUSENUMBER	VARCHAR2 (4 BYTE)
*	ZIP	VARCHAR2 (6 BYTE)
*	CITY	VARCHAR2 (80 BYTE)
*	COUNTRY	VARCHAR2 (60 BYTE)
*	COUNTRY_SHORT	CHAR (2 BYTE)
*	EMAILADDRESS	VARCHAR2 (100 BYTE)
U *	EMAIL_PROVIDER	VARCHAR2 (15 BYTE)
U *	PASSWORD	VARCHAR2 (30 BYTE)
*	GENDER	VARCHAR2 (20 BYTE)
*	GENDER_SHORT	CHAR (1 BYTE)
*	BIRTHDAY_DATE	DATE
*	BIRTHDAY_JULIAN	NUMBER (10)
	PREFERENCE1	VARCHAR2 (50 BYTE)
	PREFERENCE2	VARCHAR2 (50 BYTE)
	PREFERENCE3	VARCHAR2 (50 BYTE)
	BIRTHDAY_YEAR	NUMBER (4)
 CUSTOMER_PK (USERID)  CUSTOMER_EMAIL_PROVIDER_PASSWORD_UNIQUE		

Abbildung 5 Tabelle Customer

4.1.3 Purchases

Die Tabelle Purchases beinhaltet alle von dem Benutzer getätigten Käufe. Bei einem Kauf wird das Kaufdatum (TIMESTAMP), die Anzahl der Tickets (AMOUNT_OF_TICKETS), der Preis für jedes Ticket (PAYED_PRICE_PER_TICKET) und der gezahlte Gesamtpreis (PAYED_PRICE_TOTAL) gespeichert. Diese beiden



LABOR_HARZ_007.PURCHASES		
F *	EVENTID	NUMBER
F *	USERID	NUMBER
*	TIMESTAMP	DATE
*	AMOUNT_OF_TICKETS	NUMBER (1)
*	PAYED_PRICE_PER_TICKET	NUMBER (10,2)
	PAYED_PRICE_PER_TICKET_CENT	NUMBER (10)
	PAYED_PRICE_TOTAL	NUMBER (10,2)
	PAYED_PRICE_TOTAL_CENT	NUMBER (10)
*	PAYMETHOD	VARCHAR2 (50 BYTE)
	TICKETTYPE	VARCHAR2 (100 BYTE)
	PRICE_ADVANTAGE_EURO_TOTAL	NUMBER (10,2)
	PRICE_ADVANTAGE_CENT_TOTAL	NUMBER (10)
 SYS_C0063830 (EVENTID)  SYS_C0063831 (USERID)		

Abbildung 6 Tabelle Purchases

Preise werden jeweils noch einmal in Cent-Beträgen aufgelistet (PAYED_PRICE_TOTAL_CENT, PAYED_PRICE_PER_TICKET_CENT). Außerdem wird die gewählte Bezahlmethode (PAYMETHOD), die Art der gekauten Tickets (TICKETTYPE) und der gewährte Rabatt (PRICE_ADVANTAGE_EURO_TOTAL, PRICE_ADVANTAGE_CENT_TOTAL) gespeichert.

4.1.4 Ratings

In einer RATINGS-Tabelle wird die vom Benutzer (USERID) vergebene Bewertung (RATING) für ein bestimmtes Event (EVENTID) mit dem Bewertungsdatum (TIMESTAMP) gespeichert.



LABOR_HARZ_007.RATINGS		
F *	USERID	NUMBER
F *	EVENTID	NUMBER
*	RATING	NUMBER (1)
*	TIMESTAMP	DATE
 SYS_C0065113 (USERID)  SYS_C0065114 (EVENTID)		

Abbildung 7 Tabelle Ratings

4.1.5 Hilfstabellen

Hilfstabellen sind Tabellen, die in keiner Relation zu anderen Tabellen stehen. Der Inhalt der Tabellen wird allerdings für die Generierung weiterer Datensätze anderer Tabellen verwendet.

4.1.5.1 Countries

In der Countries-Tabelle sind Ländercodes (CODE) und deren deutschen Bedeutung (DE) hinterlegt (z.B. AT = Österreich).


LABOR_HARZ_007.COUNTRIES		
P *	CODE	CHAR (2 BYTE)
*	DE	VARCHAR2 (100 BYTE)
 COUNTRIES_PK (CODE)		

Abbildung 8 Tabelle Countries

4.1.5.2 Paymethod

Die Tabelle Paymethod beinhaltet alle Namen von möglichen Bezahlmethoden, zwischen denen der Benutzer beim Kauf eines Tickets wählen (NAME).


LABOR_HARZ_007.PAYMETHOD		
P *	NAME	VARCHAR2 (50 BYTE)
 PAYMETHOD_PK (NAME)		

Abbildung 9 Tabelle Paymethod

4.1.5.3 Tickettypes

Tickettypes beinhaltet alle möglichen Arten von Tickets und deren Rabatt. Dies beinhaltet den Namen des Tickettypen (NAME) und den damit verbundenen Preisnachlass in Prozent (PERCENTAGE).



LABOR_HARZ_007.TICKETTYPES		
P *	NAME	VARCHAR2 (100 BYTE)
	PERCENTAGE	NUMBER (2)
 SYS_C0060922 (NAME)		
 SYS_C0060922 (NAME)		

Abbildung 10 Tabelle Tickettypes

4.1.6 Analysis_Queries

In der Tabelle Analysis_Queries sind verschiedene Analyse-Abfragen (SQLQUERY) unter einem Namen (NAME) im JSON-Format abgelegt. Um eine solche Analyse auszuführen, wird die gewünschte SQL-Anfrage aus der Tabelle geholt und auf der Datenbank ausgeführt. Diese liefert dann eine Analyse mit stets aktuellen Daten, welche daraufhin visualisiert werden können.

LABOR_HARZ_007.ANALYSIS_QUERIES	
SQLQUERY	VARCHAR2 (1500 BYTE)
NAME	VARCHAR2 (100 BYTE)

Abbildung 11 Tabelle Analysis_Queries

4.1.7 Times

Die Tabelle Times enthält pro Datensatz ein Datum und all dessen abgeleiteten Eigenschaften, wie z.B. das Jahr, der Monat, den Tag der Woche / Monat / Jahr, die Kalenderwoche, das Quartal u.v.m.:

LABOR_HARZ_007.TIMES	
P * JULIANID	NUMBER
Date	DATE
Month	VARCHAR2 (10 BYTE)
Month_Short	CHAR (3 BYTE)
Month_Number	NUMBER (2)
Day_of_Week	VARCHAR2 (10 BYTE)
Day_Short	VARCHAR2 (10 BYTE)
Year	NUMBER (4)
Quarter	NUMBER (1)
Quarter_Year	VARCHAR2 (6 BYTE)
Week_number	NUMBER (2)
Day_in_Year	VARCHAR2 (10 BYTE)
Day_in_Month	NUMBER (2)
Day_in_Week	NUMBER (1)
Week_of_Month	NUMBER (2)
Month_Year	VARCHAR2 (10 BYTE)
TIMES_PK (JULIANID)	

Abbildung 12 Tabelle Times

4.2 Sequenzen

Eine Sequenz erzeugt numerische Werte, welche automatisch mit einer definierten Schrittweite inkrementiert werden. Diese erzeugten Werte werden häufig für eindeutige Primärschlüsselwerte verwendet. Dabei wird in der Regel eine Sequenz für eine einzige Tabelle verwendet. Bei der Erzeugung eines neuen Sequenzwertes wird die Sequenz automatisch inkrementiert, unabhängig eines Commits oder Rollbacks einer Transaktion [2].

Eine *Transaktion* ist eine Sammlung von SQL-Statements, die gemeinsam ausgeführt werden. Falls alle Statements ohne Fehler beendet werden, wird das Ergebnis mittels eines *Commits* in die Datenbank übertragen. Beim Auftreten eines Fehlers wird mit einem *Rollback* die Änderungen der zuvor ausgeführten Statements rückgängig gemacht und die Datenbank bleibt unverändert.

Nach Erstellung einer Sequenz kann über den vergebenen Namen und des SQL-Statements *CURRVAL* der aktuelle Wert der Sequenz ermittelt werden, wohingegen mittels *NEXTVAL* ein neuer Wert erzeugt und zurückgegeben wird.

Die Datenbank des Event-Shops beinhaltet mehrere Sequenzen, die für die Erzeugung von Primärschlüsseln verwendet werden. Jede Dimensionstabelle hat eine eigene Sequenz mit einem entsprechenden Namen. Beim Einfügen eines Datensatzes wird mittels der zu der Tabelle zugehörigen Sequenz die eindeutige ID generiert, welche zugleich als Primärschlüssel dient.

4.3 Verwendetes DWH-Schema

Das Event-Shop-DWH verwendet das Star-Schema. Die Käufe der Benutzer (Purchases-Tabelle) dienen als Faktentabelle und die Events, Customer, Tickettype, Paymethod und das Datum des Kaufes repräsentieren die Dimensionstabellen. In der *Abbildung 13 Star-Schema der Tabellen* sind diese Tabellen nach dem Sternschema angeordnet:

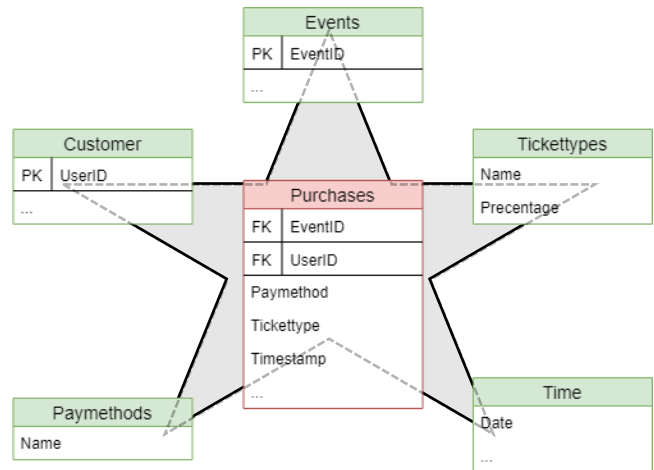


Abbildung 13 Star-Schema der Tabellen

5. ZUSAMMENFASSUNG

In dieser Facharbeit wurden die Grundlagen von Datenbanken, des Data-Warehouses und der verschiedenen Schemas erläutert. Zudem wurden die Anforderungen an die Event-Shop-Datenbank vorgestellt und die daraus erstellten Tabellen erklärt. Außerdem wurden Sequenzen und ihre Rolle in der erstellten Datenbank dem Leser nähergebracht. Abschließend wurde erläutert, welches DWH-Schema für den Event-Shop verwendet wurde und welche Tabellen als Fakten- und welche als Dimensionstabellen fungieren.

6. AUSBLICK

Für große und weitreichende Analysen sind die Datensätze in der Event-Shop-Datenbank zu gering. Zudem müssten die Tabellen für performante Analysen noch die Zeitunterteilung aus der Tabelle *Times* übernehmen und beispielsweise der Zeitstempel noch auf Jahr, Monat, Tag, Quartal, Kalenderwoche und mehr erweitert werden. Dies erspart Berechnungszeit. Allerdings ist dies in dieser Facharbeit nicht das primäre Ziel gewesen.

Außerdem könnte noch bei Analysen eine Unterteilung der Tabellen *Purchases* und *Ratings* in Data-Cubes erfolgen. Diese bilden aus den Daten sog. mehrdimensionale Würfel, wobei eine Dimension jeweils ein Produkt, Kunde oder Zeit darstellt.

7. QUELLENVERWEIS

- [1] Schneider, K. *Datenbanksysteme 2 (ehem. Data Warehouse)*. Hochschule Harz, (2019).
- [2] Lorentz, D., Oracle® Database SQL Language Reference, *CREATE SEQUENCE*, 11g Release 1 (11.1) B28286-07, *Oracle and/or its affiliates*, pages 1105-1108 (2013). [online] Available at: https://docs.oracle.com/cd/B28359_01/server.111/b28286.pdf

